Lossless and Near-Lossless Source Coding for Multiple Access Networks

Qian Zhao, *Member, IEEE*, and Michelle Effros, *Member, IEEE*

***Abstract*—A multiple access source code (MASC) is a source code designed for the following network configuration: a pair of correlated information sequences $\{X_i\}_{i=1}^{\infty}$ and $\{Y_i\}_{i=1}^{\infty}$ is drawn independent and identically distributed (i.i.d.) according to joint probability mass function (p.m.f.) $p(x, y)$; the encoder for each source operates without knowledge of the other source; the decoder jointly decodes the encoded bit streams from both sources. The work of Slepian and Wolf describes all rates achievable by MASCs of infinite coding dimension ($n \to \infty$) and asymptotically negligible error probabilities ($P_e^{(n)} \to 0$). In this paper, we consider the properties of optimal instantaneous MASCs with finite coding dimension ($n < \infty$) and both lossless ($P_e^{(n)} = 0$) and near-lossless ($P_e^{(n)} \to 0$) performance. The interest in near-lossless codes is inspired by the discontinuity in the limiting rate region at $P_e^{(n)} = 0$ and the resulting performance benefits achievable by using near-lossless MASCs as entropy codes within lossy MASCs. Our central results include generalizations of Huffman and arithmetic codes to the MASC framework for arbitrary $p(x, y)$, n, and $P_e^{(n)}$ and polynomial-time design algorithms that approximate these optimal solutions.**

***Index Terms*—Arithmetic codes, distributed source coding, Huffman codes, lossless coding, near-lossless coding, network source coding, polynomial complexity, Slepian–Wolf, Wyner–Ziv codes.**

I. Introduction

A MULTIPLE access network comprises multiple transmitters sending information to a single receiver. One example of a multiple access system is a sensor network, where separately located sensors send correlated information to a central processing unit.

Multiple access source codes (MASCs) (also known as Slepian–Wolf or distributed source codes) yield efficient data representations for multiple access systems when cooperation among the transmitters is not possible. In the MASC configuration shown in Fig. 1(a), two encoders independently describe information to a single decoder. The decoder uses the received pair of descriptions to reconstruct the original data sequences. In [1], Slepian and Wolf describe all rate pairs achievable with coding dimension $n \to \infty$ and probability of decoding error $P_e^{(n)} \to 0$ (see Fig. 1(b)).

Manuscript received June 23, 2001; revised October 2, 2002. This work was supported by NSF under Awards CCR-9909026 and CCR-0220039 and by the Caltech Lee Center for Advanced Networking. The material in this paper was presented in part at the Data Compression Conference, Snowbird, UT, March 2001 and the IEEE International Symposium on Information Theory, Washington, DC, June 2001.

The authors are with the Department of Electrical Engineering, California Institute of Technology, Pasadena, CA 91125 USA (e-mail: qianz@z.caltech.edu; effros@z.caltech.edu).

Communicated by R. Zamir, Associate Editor for Source Coding.

Digital Object Identifier 10.1109/TIT.2002.806145

This paper treats instantaneous MASCs ($n < \infty$) for both the lossless ($P_e^{(n)} = 0$) and "near-lossless" ($P_e^{(n)} \to 0$) cases. The discontinuity in the limiting rate region at $P_e^{(n)} = 0$ [2][1] motivates the interest in near-lossless coding. For finite n, this discontinuity occurs at

$$P_e^{(n)} = \min\{p^n(x^n, y^n): p^n(x^n, y^n) > 0\}$$

rather than $P_e^{(n)} = 0$. Given their superior rate capabilities, near-lossless codes are useful where small error probabilities are acceptable (e.g., as entropy codes in lossy MASCs).

Prior work on lossless instantaneous MASCs for $n < \infty$ focuses primarily on the special case of a *side-information source code* (SISC), where the decoder knows Y and the goal is to uniquely describe X using the smallest possible average rate. Work on SISC design appears in [2], [3]; these algorithms are suboptimal by [4]. Work on properties of optimal SISCs includes [5], which uses a graph-theoretic framework to derive bounds on the minimal expected rate in terms of the graph entropy, and [4], which describes necessary and sufficient conditions for the existence of a code with a given set of codeword lengths when the alphabet size of Y is two [4]. A design algorithm for lossless SISCs and MASCs for sources X and Y guaranteed to meet a maximal Hamming distance constraint appears in [6], [7].

Near-lossless codes are a special case of lossy codes with a Hamming distortion measure and an asymptotically negligible distortion. Prior work on lossy MASCs appears in [8], [9], [6], [7], [10]. Since the submission of this paper, several authors have also tackled the near-lossless coding problem using turbo codes (see, for example, [11]–[13]).

This paper derives properties of optimal MASCs and uses these properties to extend the definitions of Huffman and arithmetic codes to achieve corresponding lossless and near-lossless MASCs, giving the first constructive algorithm for building optimal lossless and near-lossless instantaneous SISCs and MASCs for general sources. The definitions and methods apply to arbitrary discrete-alphabet sources. While the encoding and decoding complexities of the proposed optimal MASCs are comparable to the corresponding complexities for traditional (single-sender, single-receiver) Huffman and arithmetic codes, the design complexities for the optimal MASCs are high. Since high design complexities seem to be unavoidable (in [14], Koulgi *et al.* show that even the lossless SISC design problem

[1]For example, if $p(x, y) > 0$ for all $(x, y) \in \mathcal{X} \times \mathcal{Y}$, then achieving $P_e^{(n)} = 0$ requires $R_X + R_Y \geq H(X) + H(Y)$ for all n. In contrast, $R_X + R_Y \geq H(X, Y)$ is achievable when $n \to \infty$ and $P_e^{(n)} \to 0$.

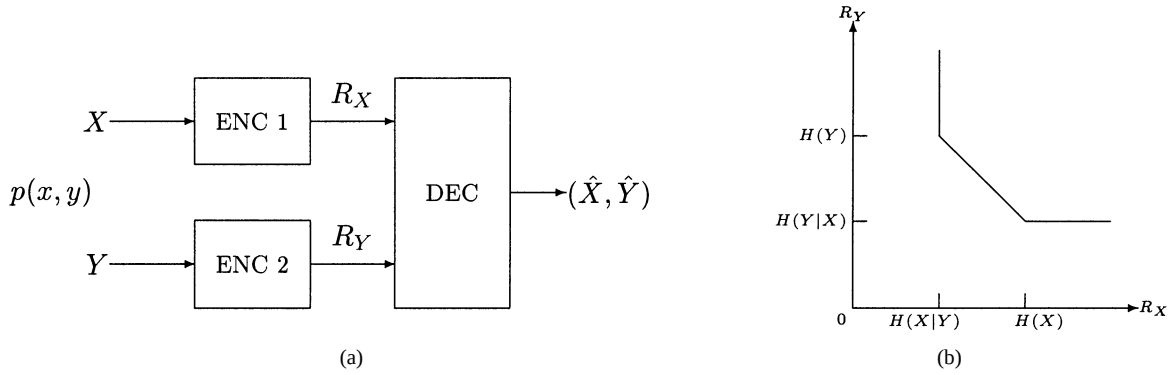


Fig. 1. (a) An MASC. (b) The Slepian–Wolf achievable rate region for MASCs.

is NP-hard), we also consider low-complexity approximate solutions. Parts of the description given here appear in [15]–[18].

The remainder of the paper is organized as follows. Section II contains the generalization of the Huffman and arithmetic code design algorithms to the lossless SISC problem. Section III gives the extension to general MASCs. Section IV treats the near-lossless MASC problem. We consider a polynomial-time design algorithm in Section V. Section VI contains experimental results. Section VII gives a summary of the key contributions of the paper.

II. Lossless Side-Information Source Codes

A. Problem Statement

Let X and Y be memoryless sources with joint probability mass function (p.m.f.) $p(x, y)$ on finite alphabet $\mathcal{X} \times \mathcal{Y}$. We use $p_X(x)$ and $p_Y(y)$ to denote the marginals of $p(x, y)$ with respect to X and Y, dropping the subscripts when they are clear from the argument to give $p(x) = p_X(x)$ and $p(y) = p_Y(y)$. A *lossless instantaneous MASC* for joint source (X, Y) consists of two encoders $\gamma_X\colon \mathcal{X} \to \{0, 1\}^\star$ and $\gamma_Y\colon \mathcal{Y} \to \{0, 1\}^\star$ and a decoder $\gamma^{-1}\colon \{0, 1\}^\star \times \{0, 1\}^\star \to \mathcal{X} \times \mathcal{Y}$. Here, $\gamma_X(x)$ and $\gamma_Y(y)$ are the binary descriptions of x and y, and the probability of decoding error is

$$P_e = \Pr(\gamma^{-1}(\gamma_X(X), \gamma_Y(Y)) \neq (X, Y)).$$

This section treats lossless coding, where $P_e \equiv 0$. Further, we concentrate exclusively on instantaneous codes, where for any input sequences $x_1, x_2, x_3, \ldots$ and $y_1, y_2, y_3, \ldots$ with $p(x_1, y_1) > 0$ the decoder reconstructs (x_1, y_1) by reading only the first $|\gamma_X(x_1)|$ bits from $\gamma_X(x_1)\gamma_X(x_2)\gamma_X(x_3)\ldots$ and the first $|\gamma_Y(y_1)|$ bits from $\gamma_Y(y_1)\gamma_Y(y_2)\gamma_Y(y_3)\ldots$ (without prior knowledge of these lengths).

When Y is perfectly known to the decoder, the problem reduces to the SISC problem. This scenario describes MASCs where γ_Y encodes Y using a traditional code for p.m.f. $p(y)$ so that γ_X can encode X assuming that the decoder knows Y. In this case, $\gamma^{-1}\colon \{0, 1\}^\star \times \mathcal{Y} \to \mathcal{X}$. If the decoder can correctly reconstruct x_1 by reading only the first $|\gamma_X(x_1)|$ bits of $\gamma_X(x_1)\gamma_X(x_2)\gamma_X(x_3)\ldots$, then (γ_X, γ^{-1}) is a *lossless instantaneous SISC*. We wish to design a lossless instantaneous SISC that achieves the lowest possible expected rate. We treat code design for general MASCs in Section III.

Lemma 1 (SISC Prefix Property): Code γ_X is a lossless instantaneous SISC for X given Y if and only if for each x, x', y with $p(x, y) > 0$ and $p(x', y) > 0$, $\{\gamma_X(x), \gamma_X(x')\}$ is prefix free.

Proof: Necessary: If there exists some $y \in \mathcal{Y}$ and $x, x' \in \mathcal{X}$ for which $p(x, y) > 0$, $p(x', y) > 0$, and $\gamma_X(x)$ is a prefix of $\gamma_X(x')$, then the codewords for x and x' cannot be instantaneously distinguished when $Y = y$. Sufficient: The decoder receives Y and performs the mapping

$$\gamma^{-1}(\cdot, Y)\colon \{0, 1\}^\star \to \{x \in \mathcal{X}\colon p(x, Y) > 0\}.$$

Since $\{\gamma_X(x)\colon\ p(x, Y) > 0\}$ is prefix free, the code is instantaneous. □

Distinct symbols $x, x' \in \mathcal{X}$ are *confusable* under $p(x, y)$, written $x \fallingdotseq x'$, if $p(x, y) > 0$ and $p(x', y) > 0$ for some $y \in \mathcal{Y}$. By Lemma 1, an SISC's description of x and x' can be identical ($\gamma_X(x) = \gamma_X(x')$) or the description of x can be a proper prefix of the description of x' (written $\gamma_X(x) \prec \gamma_X(x')$) if the symbols are *not* confusable ($x \not\fallingdotseq x'$)—that is, if knowing Y eliminates any ambiguity between the descriptions of x and x'. The design algorithm in [3] allows $\gamma_X(x) = \gamma_X(x')$ when $x \not\fallingdotseq x'$ but never allows $\gamma_X(x) \prec \gamma_X(x')$, giving a code consistent with the *unrestricted inputs* codes of [5]. While [5] and [4] do not treat code design, both address the two types of prefix violations. The discussions in [2] and [5] associate with each p.m.f. $p(x, y)$ on $\mathcal{X} \times \mathcal{Y}$ a graph $G = (\mathcal{X}, E_{\mathcal{X}})$, where there is an edge between $x, x' \in \mathcal{X}$ if and only if $x \neq x'$ and $x \fallingdotseq x'$. Alon and Orlitsky state that code γ_X is valid if and only if for every edge $\{x, x'\} \in E_{\mathcal{X}}$, $\{\gamma_X(x), \gamma_X(x')\}$ satisfies the prefix condition; thus, a valid code from [5] is a lossless instantaneous SISC. Since the set of uniquely decodable codes on colorings of G is a subset of the set of valid codes, they bound the expected rate of a side-information code as a function of the "chromatic entropy" of G. While this approach yields elegant rate bounds, the difference between the resulting expected rate and the optimal performance can be arbitrarily large [5]. Building on the results of [5], the recent work of [19] characterizes the asymptotic rate of an SISC as the complementary graph entropy of graph G.

B. Groups, Partitions, and Matched Codes

While graphs give a strong intuition into the performance bounds of [5], we find them difficult to work with for *optimal*

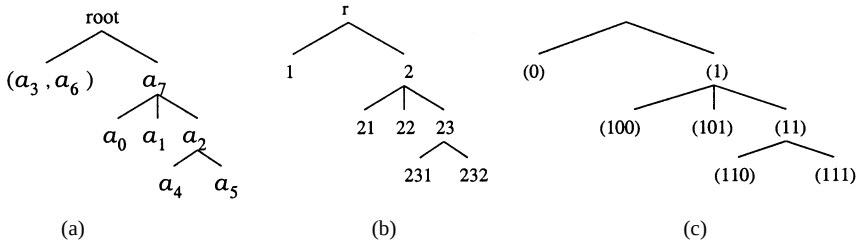


Fig. 2. (a) Partition tree $\mathcal{T}(\mathcal{P}(\mathcal{X}))$. (b) Labels for $\mathcal{T}(\mathcal{P}(\mathcal{X}))$. (c) Matched code for $\mathcal{P}(\mathcal{X})$.

code design; we, therefore, turn instead to tree structures in the discussion that follows. We use trees to illustrate the prefix relationships between codewords: $\gamma_X(x) \prec \gamma_X(x')$ if and only if x is an ancestor of x' in the corresponding tree, and $\gamma_X(x) = \gamma_X(x')$ if and only if x and x' occupy the same node of the corresponding tree. The resulting trees are similar to Huffman code trees in that all symbols descending from a common parent have descriptions that share a common prefix; they differ from Huffman trees in that they need not be binary, symbols can reside at internal nodes as well as leaves, and multiple symbols can occupy the same node. We call each possible subtree a "group"; the number of levels in a group equals the number of levels in the corresponding tree. Precise definitions follow; these definitions rule out any construction that cannot yield a lossless instantaneous SISC.

The collection $\mathcal{G} = (x_1, \ldots, x_m)$ is a legitimate *one-level group* for $p(x, y)$ if for any distinct $x_i, x_j \in \mathcal{G}$, $x_i \not\succ x_j$.[2] The *tree representation* $\mathcal{T}(\mathcal{G})$ for one-level group $\mathcal{G}$ is a single node representing all members of $\mathcal{G}$. For the p.m.f. in Table IV(a) in the Appendix, (a_0), (a_4, a_7), and (a_0, a_4, a_7) are all examples of legitimate one-level groups.

A *two-level group* for $p(x, y)$, denoted by $\mathcal{G} = (\mathcal{R}: \mathcal{C}(\mathcal{R}))$, comprises a root $\mathcal{R}$ and its children $\mathcal{C}(\mathcal{R})$; $\mathcal{R}$ is a one-level group, $\mathcal{C}(\mathcal{R})$ is a set of one-level groups, and $\mathcal{G}' \not\succ \mathcal{R}$ for all $\mathcal{G}' \in \mathcal{C}(\mathcal{R})$, where for any groups $\mathcal{G}_1$ and $\mathcal{G}_2$, $\mathcal{G}_1 \not\succ \mathcal{G}_2$ if and only if $x_1 \not\succ x_2$ for all $x_1 \in \mathcal{G}_1$ and $x_2 \in \mathcal{G}_2$. Members of all $\mathcal{G}' \in \mathcal{C}(\mathcal{R})$ are called members of $\mathcal{C}(\mathcal{R})$, and members of $\mathcal{R}$ and $\mathcal{C}(\mathcal{R})$ are called members of $\mathcal{G}$. In the tree representation $\mathcal{T}(\mathcal{G})$ for $\mathcal{G}$, $\mathcal{T}(\mathcal{R})$ is the root of $\mathcal{T}(\mathcal{G})$ and the parent of all subtrees $\mathcal{T}(\mathcal{G}')$ for $\mathcal{G}' \in \mathcal{C}(\mathcal{R})$. An example of a two-level group for the p.m.f. in Table IV(a) is $\mathcal{G}_2 = ((a_4)\colon \{(a_0), (a_2, a_7), (a_6)\})$. In this case, $\mathcal{R} = (a_4)$ and $\mathcal{C}(\mathcal{R}) = \{(a_0), (a_2, a_7), (a_6)\}$. The members of $\mathcal{C}(\mathcal{R})$ are $\{a_0, a_2, a_6, a_7\}$; the members of $\mathcal{G}_2$ are $\{a_0, a_2, a_4, a_6, a_7\}$. The tree representation $\mathcal{T}(\mathcal{G}_2)$ is a two-level tree comprising a root and its three children, each of which is a single node.

For each subsequent $M > 2$, an *M-level group* for $p(x, y)$ is a pair $\mathcal{G} = (\mathcal{R}{:}\mathcal{C}(\mathcal{R}))$ such that $\mathcal{G}' \not\succ \mathcal{R}$ for all $\mathcal{G}' \in \mathcal{C}(\mathcal{R})$. Here, $\mathcal{R}$ is a one-level group and $\mathcal{C}(\mathcal{R})$ is a set of groups of $M-1$ or fewer levels, at least one of which has $(M-1)$ levels. The members of $\mathcal{R}$ and $\mathcal{C}(\mathcal{R})$ together comprise the members of $\mathcal{G} = (\mathcal{R}\colon \mathcal{C}(\mathcal{R}))$. Again, $\mathcal{T}(\mathcal{R})$ is the root of $\mathcal{T}(\mathcal{G})$ and the parent of all subtrees $\mathcal{T}(\mathcal{G}')$ for $\mathcal{G}' \in \mathcal{C}(\mathcal{R})$. For any $M > 1$, an M-level group is also called a *multilevel group*. An example of a three-level group for the p.m.f. in Table IV(a) is $\mathcal{G}_3 = ((a_7)\colon \{(a_0), (a_1), ((a_2)\colon \{(a_4), (a_5)\})\})$. In $\mathcal{T}(\mathcal{G}_3)$, the root $\mathcal{T}(a_7)$ of the three-level group has three children: the first two children are nodes $\mathcal{T}(a_0)$ and $\mathcal{T}(a_1)$; the third child is a two-level tree with root node $\mathcal{T}(a_2)$ and children $\mathcal{T}(a_4)$ and $\mathcal{T}(a_5)$.

A *partition* $\mathcal{P}(\mathcal{X})$ on $\mathcal{X}$ for p.m.f. $p(x, y)$ is a complete and nonoverlapping set of groups. That is, $\mathcal{P}(\mathcal{X}) = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_m\}$ satisfies $\bigcup_{i=1}^{m} \mathcal{G}_i = \mathcal{X}$ and $\mathcal{G}_j \bigcap \mathcal{G}_k = \phi$ for any $j \neq k$, where each $\mathcal{G}_i \in \mathcal{P}(\mathcal{X})$ is a group for $p(x, y)$, and $\mathcal{G}_j \cup \mathcal{G}_k$ and $\mathcal{G}_j \cap \mathcal{G}_k$ refer to the union and intersection, respectively, of the members of $\mathcal{G}_j$ and $\mathcal{G}_k$. The tree representation of a partition is called a *partition tree*. The partition tree $\mathcal{T}(\mathcal{P}(\mathcal{X}))$ for partition $\mathcal{P}(\mathcal{X}) = \{\mathcal{G}_1, \mathcal{G}_2, \ldots, \mathcal{G}_m\}$ has an empty root $\boldsymbol{r}$ with m children, $\mathcal{T}(\mathcal{G}_1), \ldots, \mathcal{T}(\mathcal{G}_m)$. A partition tree is not necessarily a regular k-ary tree since the number of descendants varies from one node to the next. Fig. 2(a) gives a partition tree for partition $\mathcal{P}(\mathcal{X}) = \{(a_3, a_6), \mathcal{G}_3\}$.

For any one-level group $\mathcal{G}$ at depth d in $\mathcal{T}(\mathcal{P}(\mathcal{X}))$, let $\boldsymbol{n}$ describe the d-step path from root $\boldsymbol{r}$ to node $\mathcal{T}(\mathcal{G})$ in $\mathcal{T}(\mathcal{P}(\mathcal{X}))$. We often refer to $\mathcal{G}$ by describing this path. Thus, $\mathcal{T}(\boldsymbol{n}) = \mathcal{T}(\mathcal{G})$. For notational simplicity, we sometimes substitute $\boldsymbol{n}$ for $\mathcal{T}(\boldsymbol{n})$ when it is clear from the context that we are talking about the node rather than the one-level group at that node (e.g., $\boldsymbol{n} \in \mathcal{T}(\mathcal{P}(\mathcal{X}))$ rather than $\mathcal{T}(\boldsymbol{n}) \in \mathcal{T}(\mathcal{P}(\mathcal{X}))$). To make the path descriptions unique, we fix an order on the descendants of each node and number them from left to right. Thus, $\boldsymbol{n}$'s children are labeled as $\boldsymbol{n}1, \boldsymbol{n}2, \ldots, \boldsymbol{n}K(\boldsymbol{n})$, where $\boldsymbol{n}k$ is a vector created by concatenating k to $\boldsymbol{n}$ and $K(\boldsymbol{n})$ is the number of children descending from $\boldsymbol{n}$. The labeled partition tree for Fig. 2(a) appears in Fig. 2(b).

The *node probability* $q(\boldsymbol{n})$ of one-level group $\boldsymbol{n}$ is the sum of the probabilities of that group's members. The *subtree probability* $Q(\boldsymbol{n})$ of one-level group $\boldsymbol{n}$ is the sum of probabilities of $\boldsymbol{n}$'s members and descendants in $\mathcal{T}(\mathcal{P}(\mathcal{X}))$. In Fig. 2(b), $q(23) = p_X(a_2)$ and $Q(23) = p_X(a_2) + p_X(a_4) + p_X(a_5)$.

A *matched code* γ_X for partition $\mathcal{P}(\mathcal{X})$ is any binary code[3] such that for any node $\boldsymbol{n} \in \mathcal{T}(\mathcal{P}(\mathcal{X}))$ and symbols $x_1, x_2 \in \boldsymbol{n}$ and $x_3 \in \boldsymbol{n}k$, $k \in \{1, \ldots, K(\boldsymbol{n})\}$: 1) $\gamma_X(x_1) = \gamma_X(x_2)$; 2) $\gamma_X(x_1) \prec \gamma_X(x_3)$; 3) $\{\gamma_X(\boldsymbol{n}k)\colon k \in \{1, \ldots, K(\boldsymbol{n})\}\}$ is prefix free. (We use $\gamma_X(\boldsymbol{n})$ interchangeably with $\gamma_X(x)$ for any $x \in \boldsymbol{n}$.) If symbol $x \in \mathcal{X}$ belongs to one-level group $\mathcal{G}$, then $\gamma_X(x)$ describes the path in $\mathcal{T}(\mathcal{P}(\mathcal{X}))$ from $\boldsymbol{r}$ to $\mathcal{T}(\mathcal{G})$; the path description is a concatenated list of step descriptions, where the step from $\boldsymbol{n}$ to $\boldsymbol{n}k$, $k \in \{1, \ldots, K(\boldsymbol{n})\}$, is described using a prefix code on $\{1, \ldots, K(\boldsymbol{n})\}$.

An example of a matched code for the partition of Fig. 2(a) appears in Fig. 2(c). Fig. 3 shows that code's encoder and de-

[2]All symbols given the same color in a coloring in [5] are a one-level group. Thus, any independent set in the graph G can be a one-level group.

[3]We here focus on codes with binary channel alphabet $\{0, 1\}$. The extension to codes with other finite channel alphabets is straightforward.

$$\gamma_X(x) = \begin{cases} 0 & \text{if } x \in \{a_3, a_6\} \\ 1 & \text{if } x = a_7 \\ 100 & \text{if } x = a_0 \\ 101 & \text{if } x = a_1 \\ 11 & \text{if } x = a_2 \\ 110 & \text{if } x = a_4 \\ 111 & \text{if } x = a_5 \end{cases}$$

$\gamma^{-1}((\gamma_X(x_1)\ldots), y)$								
$\gamma_X(x_1)\ldots \backslash y$	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
$0\ldots$	$-$	a_3	a_6	a_3	$-$	a_6	a_6	a_6
$1\ldots$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	a_7	a_7
$10\ldots$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\downarrow$	$\uparrow$	$\uparrow$
$100\ldots$	a_0	$-$	a_0	a_0	$-$	$-$	$\uparrow$	$\uparrow$
$101\ldots$	$-$	a_1	$-$	$-$	a_1	a_1	$\uparrow$	$\uparrow$
$11\ldots$	a_2	$\downarrow$	a_2	$\downarrow$	$\downarrow$	$\downarrow$	$\uparrow$	$\uparrow$
$110\ldots$	$\uparrow$	a_4	$\uparrow$	$-$	a_4	$-$	$\uparrow$	$\uparrow$
$111\ldots$	$\uparrow$	$-$	$\uparrow$	a_5	a_5	a_5	$\uparrow$	$\uparrow$

Fig. 3. The encoder and decoder for the example in Fig. 2. For the decoder, $-$ marks an event that can never occur, $\downarrow$ marks a case where the decoder knows it has not reached the end of $\gamma_X(x)$, and $\uparrow$ marks a case where the decoder would have decoded based on fewer symbol than are given.

coder. In the decoder definition "$-$" marks an event that can never occur, "$\downarrow$" marks a case where the decoder knows it has not reached the end of $\gamma_X(x)$, and "$\uparrow$" marks a case where the decoder would have decoded based on fewer symbol than are given. For example, since only $\gamma_X(a_3)$ and $\gamma_X(a_6)$ begin with $0\cdots$ and $p(a_3, a_0) = p(a_6, a_0) = 0$, the decoder never receives $\gamma_X(x) = 0\cdots$ when $Y = a_0$; if the decoder receives binary string $11\cdots$ when $Y = a_1$, then the decoder has not reached the end of $\gamma_X(x_1)$ since $p(a_7, a_1) = p(a_2, a_1) = 0$ and every other $\gamma_X(x_1)$ beginning $11\cdots$ has length greater than 2; if the decoder receives binary string $111\cdots$ when $Y = a_0$, then it decodes to a_2 after only 2 bits. The code is instantaneous since the decoder can always decode after no more than $|\gamma_X(x)|$ bits. The code is lossless since $\gamma^{-1}(\gamma_X(x), y) = x$ whenever $p(x, y) > 0$ and probability 0 events cannot occur. The code achieves expected rate

$$E_X|\gamma_X(X)| = 2.12 < H(X) = 2.91$$

by violating Kraft's inequality. (Kraft's inequality does not apply to SISCs [4].) This rate may be further reduced if we choose the partition and its matched code more carefully. For example, setting $\gamma_X(a_0) = \gamma_X(a_1) = 10$ in this code gives a lossless instantaneous code with lower expected rate.

C. All Lossless Codes are Matched Codes

In the above framework, a partition specifies the prefix and equivalence relationships in the binary descriptions of $x \in \mathcal{X}$; a matched code is any code with those properties. Theorem 1 establishes the equivalence of matched codes and lossless instantaneous SISCs.

Theorem 1: Code γ_X is a lossless, instantaneous SISC for $p(x, y)$ if and only if γ_X is a matched code for some partition $\mathcal{P}(\mathcal{X})$ for $p(x, y)$.

Proof: Forward: By the definitions of partitions and matched codes, only symbols that are not confusable can be assigned codewords that violate the prefix condition. Thus, any matched code is a lossless instantaneous code by Lemma 1.

Converse: Given γ_X, we construct a partition $\mathcal{P}(\mathcal{X})$ for $p(x, y)$ such that γ_X is a matched code for $\mathcal{P}(\mathcal{X})$. We begin by building a binary tree $\mathcal{T}_2$ with symbol x at the node reached by following path $\gamma_X(x)$ downward from the tree root. We build partition tree $\mathcal{T}$ from binary tree $\mathcal{T}_2$ by visiting the nodes of $\mathcal{T}_2$ one by one and modifying them as follows. If the current node is the root of the tree and that node is occupied by some $x \in \mathcal{X}$, we add a new node $\boldsymbol{r}$ as the parent of the current node; if the current node is not the root of the tree and that node is empty, we remove the current node from the tree, attaching the node's children (if any) directly to the node's parent; otherwise, we make no change. Tree $\mathcal{T}$ is a partition tree for some partition $\mathcal{P}(\mathcal{X})$ for $p(x, y)$ since 1) γ_X lossless implies each nonempty node $\boldsymbol{n} \in \mathcal{T}$ is a legitimate one-level group; 2) γ_X instantaneous implies each subtree of $\mathcal{T}$ is a legitimate multilevel group; and 3) the root of $\mathcal{T}$ is an empty node with one or more multilevel groups descending from it. Here 1) and 2) follow from Lemma 1 while 3) is by construction given 1) and 2). Code γ_X is a matched code for $\mathcal{P}(\mathcal{X})$ since: for any $\boldsymbol{n} \in \mathcal{T}(\mathcal{P}(\mathcal{X}))$, $x_1, x_2 \in \boldsymbol{n}$, and $x_3 \in \boldsymbol{n}k$, $\gamma_X(x_1) = \gamma_X(x_2)$, $\gamma_X(x_1) \prec \gamma_X(x_3)$; and

$$\{\gamma_X(\boldsymbol{n}k)\colon k \in \{1, \ldots, K(\boldsymbol{n})\}\}$$

is prefix free by construction. □

Using Theorem 1, we break the problem of lossless SISC design into two parts: partition design and matched code design. While the choice of one-level groups in a partition design is equivalent to choosing a coloring of the nodes of graph G, the coloring corresponding to the optimal partition need not be the entropy-minimizing coloring [5]. We conjecture that both finding the optimal coloring and finding the optimal prefix relationships for that coloring are NP-hard problems,[4] and we combine these "hard" parts into the single step of partition design. We treat both optimal partition design and fast approximation

[4]The intuition behind the first conjecture results from the difficulty of coloring problems for a wide variety of applications. The intuition for the second conjecture comes from the observation that finding the optimal prefix relationship is an optimal partition design problem for the set of distributions whose optimal partitions use proper prefix relationships but do not allow $\gamma_X(x) = \gamma_X(x')$ for any $x \neq x'$. Since this restriction does not suggest any obvious constraints on $p(x, y)$, we conjecture that finding optimal prefix relationships, like optimal partition design, is NP-hard.

algorithms in later sections. Given a partition, optimal matched code design requires only polynomial time. We treat optimal matched code design next. There is no analog to matched code design in [5].

D. Matched Code Design: Optimal Huffman and Arithmetic Codes

We wish to design the optimal matched code for an arbitrary fixed partition $\mathcal{P}(\mathcal{X})$ for $p(x, y)$. In traditional lossless coding, the optimal description lengths are $l^\star(x) = -\log p(x)$ for all $x \in \mathcal{X}$ if those lengths are all integers. Theorem 2 gives the corresponding result for lossless SISCs on a fixed partition $\mathcal{P}(\mathcal{X})$.

Theorem 2: Given partition $\mathcal{P}(\mathcal{X})$ for $p(x, y)$, the optimal matched code for $\mathcal{P}(\mathcal{X})$ has description lengths $l^\star(\boldsymbol{r}) = 0$ and

$$l^\star(\boldsymbol{n}k) = l^\star(\boldsymbol{n}) - \log_2 \left(\frac{Q(\boldsymbol{n}k)}{\sum_{j=1}^{K(\boldsymbol{n})} Q(\boldsymbol{n}j)} \right)$$

for all $\boldsymbol{n} \in \mathcal{T}(\mathcal{P}(\mathcal{X}))$ and $k \in \{1, \ldots, K(\boldsymbol{n})\}$ if those lengths are all integers. Here, $l^\star(\boldsymbol{n}) = l$ implies $l^\star(x) = l$ for each symbol x in one-level group $\boldsymbol{n}$.

Proof: Given $x \in \boldsymbol{n} \in \mathcal{T}(\mathcal{P}(\mathcal{X}))$, let $l(\boldsymbol{n}) = |\gamma_X(x)|$. Then for any matched code γ_X for $\mathcal{P}(\mathcal{X})$

$$\begin{aligned} E|\gamma_X(X)| &= \sum_{\boldsymbol{n} \in \mathcal{T}(\mathcal{P}(\mathcal{X}))} q(\boldsymbol{n}) l(\boldsymbol{n}) \\ &= \sum_{\boldsymbol{n} \in \mathcal{T}(\mathcal{P}(\mathcal{X}))} \sum_{k=1}^{K(\boldsymbol{n})} Q(\boldsymbol{n}k)(l(\boldsymbol{n}k) - l(\boldsymbol{n})). \end{aligned}$$

Thus, the minimal expected rate is achieved by minimizing each sum $\sum_{k=1}^{K(\boldsymbol{n})} Q(\boldsymbol{n}k)(l(\boldsymbol{n}k) - l(\boldsymbol{n}))$ independently. If we write $\gamma_X(\boldsymbol{n}k) = gg_k$ for each $k \in \{1, \ldots K(\boldsymbol{n})\}$, where $g = \gamma_X(\boldsymbol{n})$, then g_k is the suffix associated with the kth descendant of $\boldsymbol{n}$ and $|g_k| = l(\boldsymbol{n}k) - l(\boldsymbol{n})$. For $\{g_1, \ldots, g_{K(\boldsymbol{n})}\}$ to satisfy the prefix condition, $\{|g_1|, \ldots, |g_{K(\boldsymbol{n})}|\}$ must satisfy Kraft's inequality. The minimization of $\sum_{k=1}^{K(\boldsymbol{n})} Q(\boldsymbol{n}k)|g_k|$ subject to $\sum_{k=1}^{K(\boldsymbol{n})} 2^{-|g_k|} \leq 1$ is achieved by setting

$$|g_k| = l(\boldsymbol{n}k) - l(\boldsymbol{n}) = -\log_2 \left(\frac{Q(\boldsymbol{n}k)}{\sum_{j=1}^{K(\boldsymbol{n})} Q(\boldsymbol{n}j)} \right)$$

which completes the proof. □

The proof of Theorem 2 demonstrates that we can design matched codes by designing entropy codes on the children of each internal node of a partition tree. All entropy coding algorithms are candidates for matched code design. We focus on matched Huffman and arithmetic coding. For any node $\boldsymbol{n}$ with $K(\boldsymbol{n}) > 0$, the Huffman code $\gamma^{(H)}_{X,\mathcal{P}(\mathcal{X})}$ describes the step from $\boldsymbol{n}$ to $\boldsymbol{n}k$ using a Huffman code designed for p.m.f.

$$\left\{ Q(\boldsymbol{n}k) \Big/ \sum_{j=1}^{K(\boldsymbol{n})} Q(\boldsymbol{n}j) \right\}_{k=1}^{K(\boldsymbol{n})}$$

on alphabet $\{1, \ldots, K(\boldsymbol{n})\}$. The arithmetic code $\gamma^{(A)}_{X,\mathcal{P}(\mathcal{X})}$ uses arithmetic codes matched to the same p.m.f.'s. Both of these strategies give polynomial-time algorithms. Theorem 3 proves the optimality of matched Huffman codes. Before giving that result, we work an example.

Example: In building a matched Huffman code for the partition in Fig. 2(a), we work from the top of partition tree $\mathcal{T}$. We begin by designing a Huffman code for p.m.f.

$$\left\{ Q(k) \Big/ \sum_{j=1}^{K(\boldsymbol{r})} Q(j) \right\}_{k=1}^{K(\boldsymbol{r})}$$

on the $K(\boldsymbol{r})$ descendants of the (empty) root of $\mathcal{T}$. In this case, $K(\boldsymbol{r}) = 2$ ("1" $= (a_3, a_6)$ and "2" $= (a_7)$), the p.m.f. is

$$\begin{aligned} &\{p_X(a_3) + p_X(a_6),\, p_X(a_7) + p_X(a_0) \\ &\quad + p_X(a_1) + p_X(a_2) + p_X(a_4) + p_X(a_5)\} = \{0.21,\, 0.79\} \end{aligned}$$

and the Huffman code is $\{0, 1\}$. We repeat this process for each subsequent tree node $\boldsymbol{n}$ with $K(\boldsymbol{n}) > 0$. Node 2 gives $K(2) = 3$, p.m.f.

$$\begin{aligned} &\{p_X(a_0)/Q_1,\, p_X(a_1)/Q_1,\, (p_X(a_2) + p_X(a_4) + p_X(a_5))/Q_1\} \\ &\qquad = \{0.1/Q_1,\, 0.19/Q_1,\, 0.37/Q_1\} \quad (Q_1 = 0.66) \end{aligned}$$

and Huffman code $\{00, 01, 1\}$. Node 23 gives $K(23) = 2$, p.m.f.

$$\begin{aligned} &\{p_X(a_4)/Q_2,\, p_X(a_5)/Q_2\} = \{0.11/Q_2,\, 0.06/Q_2\} \\ &\qquad (Q_2 = 0.17) \end{aligned}$$

and Huffman code $\{0, 1\}$. Finally, $\gamma_X(\boldsymbol{n})$ concatenates the Huffman codewords for all branches traversed in moving from $\boldsymbol{r}$ to $\boldsymbol{n}$ in $\mathcal{T}$. The codewords for this example appear in Fig. 2(c).

Theorem 3: Given a partition $\mathcal{P}(\mathcal{X})$, matched Huffman codes for $\mathcal{P}(\mathcal{X})$ achieve the optimal expected rate over all matched codes for $\mathcal{P}(\mathcal{X})$.

Proof: Let $\mathcal{T}$ be the partition tree of $\mathcal{P}(\mathcal{X})$. The codeword length of a node $\boldsymbol{n} \in \mathcal{T}$ is denoted by $l(\boldsymbol{n})$. The average length $\bar{l}$ for $\mathcal{P}(\mathcal{X})$ is

$$\bar{l} = \sum_{\boldsymbol{n} \in \mathcal{T}} q(\boldsymbol{n}) l(\boldsymbol{n}) = \sum_{k=1}^{K(\boldsymbol{r})} \left(Q(k) l(k) + \Delta\bar{l}(k) \right)$$

where, for each $k \in \{1, \ldots, K(\boldsymbol{r})\}$

$$\Delta\bar{l}(k) = \sum_{k\boldsymbol{n} \in \mathcal{T}} q(k\boldsymbol{n})(l(k\boldsymbol{n}) - l(k)).$$

Note that $\sum_{k=1}^{K(\boldsymbol{r})} Q(k) l(k)$ and $\{\Delta\bar{l}(k)\}$ can be minimized independently. Thus,

$$\min \bar{l} = \min \sum_{k=1}^{K(\boldsymbol{r})} Q(k) l(k) + \sum_{k=1}^{K(\boldsymbol{r})} \min \Delta\bar{l}(k).$$

In matched Huffman coding, working from the top to the bottom of the partition tree, we first minimize $\sum_{k=1}^{K(\boldsymbol{r})} Q(k) l(k)$ over all integer lengths $l(k)$ by employing Huffman codes on $Q(k)$. We then minimize each $\Delta\bar{l}(k)$ over all integer-length codes by similarly breaking each down layer by layer and minimizing the expected length at each layer. □

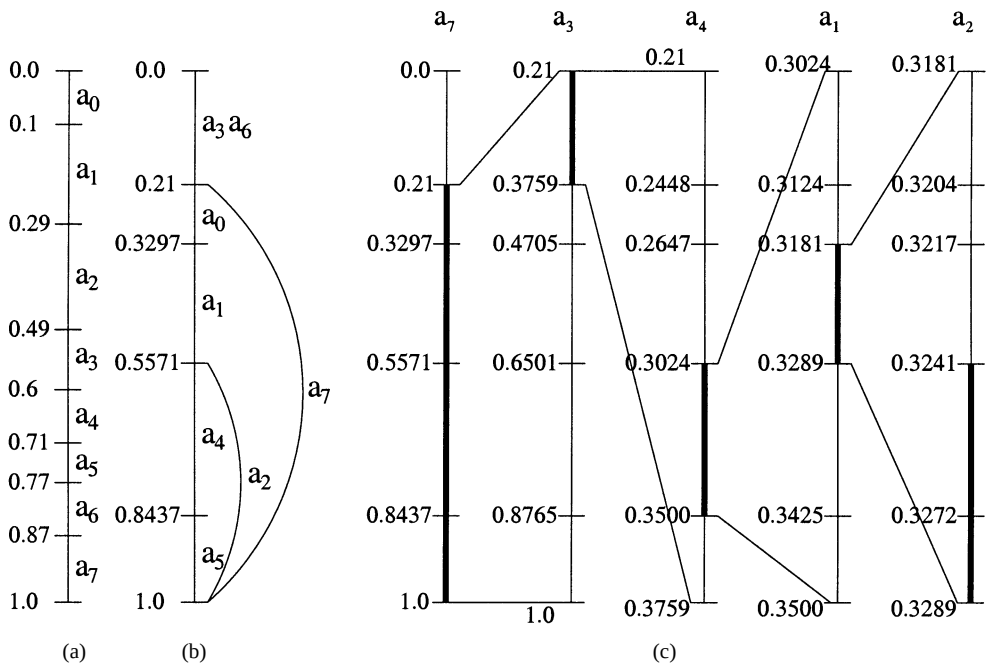


Fig. 4. Dividing the unit interval in (a) traditional arithmetic coding and (b) matched arithmetic coding for partition $\mathcal{P}(\mathcal{X})$ of Fig. 2(a). (c) Matched arithmetic coding for sequence $a_7a_3a_4a_1a_2$.

In traditional arithmetic coding (with no side information), the description length of data sequence x^n is $l(x^n) = \lceil -\log p_X(x^n) \rceil + 1$, where $p_X(x^n)$ is the probability of x^n. In designing the matched arithmetic code of x^n for a given partition $\mathcal{P}(\mathcal{X})$, we use the decoder's knowledge of y^n to decrease the description length of x^n. The following example, illustrated in Fig. 4, demonstrates the techniques of matched arithmetic coding for the partition given in Fig. 2(a).

In traditional arithmetic coding, data sequence X^n is represented by an interval of the $[0, 1)$ line. We describe X^n by describing the midpoint of the corresponding interval to sufficient accuracy to avoid confusion with neighboring intervals. We find the interval for x^n recursively, by first breaking $[0, 1)$ into intervals corresponding to all possible values of x_1 (see Fig. 4(a)), then breaking the interval for the observed X_1 into subintervals corresponding to all possible values of X_1x_2, and so on. Given the interval $A \subseteq [0, 1)$ for X^k for some $0 \leq k < n$ (the interval for X^0 is $[0, 1)$), the subintervals for $\{X^k x_{k+1}\}$ are ordered subintervals of A with lengths proportional to $p(x_{k+1})$.

In matched arithmetic coding for partition $\mathcal{P}(\mathcal{X})$, we again describe X^n by describing the midpoint of a recursively constructed subinterval of $[0, 1)$. The intervals here correspond to nodes, and we describe symbol $x \in \mathcal{X}$ by describing the midpoint of the interval corresponding to the node $\boldsymbol{n}$ for which $x \in \boldsymbol{n}$. In describing x_1, the interval for root $\boldsymbol{r}$ is $[0, 1)$ with length $p^{(A)}(\boldsymbol{r}) = 1$. We define the remainder of the intervals recursively. The interval for any $\boldsymbol{n} \in \mathcal{T}(\mathcal{P}(\mathcal{X}))$ comprises $K(\boldsymbol{n})$ ordered subintervals of lengths

$$p^{(A)}(\boldsymbol{n}k) = \left(\frac{Q(\boldsymbol{n}k)}{\sum_{k=1}^{K(\boldsymbol{n})} Q(\boldsymbol{n}k)}\right) p^{(A)}(\boldsymbol{n})$$
$$= \left(\frac{Q(\boldsymbol{n}k)}{Q(\boldsymbol{n}) - q(\boldsymbol{n})}\right) p^{(A)}(\boldsymbol{n}), \qquad k \in \{1, \ldots, K(\boldsymbol{n})\}$$

corresponding to the $K(\boldsymbol{n})$ children of $\boldsymbol{n}$ in the partition tree. The nested nature of the intervals parallels the situation in matched Huffman coding where one symbol's description is the prefix of another symbol's description. Again, for any legitimate partition $\mathcal{P}(\mathcal{X})$, the decoder can uniquely distinguish between symbols with nested intervals using its knowledge of the side information.

Refining the interval for sequence X^{i-1} to find the subinterval for X^i involves carving the current interval into subintervals of sizes proportional to those found above. We finally describe X^n by describing the center of its corresponding subinterval to an accuracy sufficient to distinguish it from its neighboring subintervals. To ensure unique decodability

$$l^{(A)}(x^n) = \left\lceil -\log p^{(A)}(x^n) \right\rceil + 1$$

where $p^{(A)}(x^n)$ is the length of the subinterval corresponding to string x^n. Given a fixed partition $\mathcal{P}(\mathcal{X})$, suppose $x \in \boldsymbol{n}(x) \in \mathcal{T}(\mathcal{P}(\mathcal{X}))$ and $\boldsymbol{n}_0(x)$ is the parent of $\boldsymbol{n}(x)$. Then

$$\begin{aligned} & l^{(A)}(x^n) \\ &= \left\lceil -\log p^{(A)}(x^n) \right\rceil + 1 \\ &= \left\lceil \sum_{i=1}^{n} -\log p^{(A)}(\boldsymbol{n}(x_i)) \right\rceil + 1 \\ &= \left\lceil \sum_{i=1}^{n} \left(-\log p^{(A)}(\boldsymbol{n}_0(x_i)) - \log \frac{Q(\boldsymbol{n}(x_i))}{\sum_{k=1}^{K(\boldsymbol{n}_0(x_i))} Q(\boldsymbol{n}_0(x_i)k)} \right) \right\rceil + 1 \\ &< \sum_{i=1}^{n} l^{\star}(x_i) + 2 \end{aligned}$$

where $l^\star()$ is the optimal length function specified in Theorem 2. Thus, the description length $l^{(A)}(x^n)$ in coding data sequence x^n using a one-dimensional "matched arithmetic code" $\gamma^{(A)}_{X,\mathcal{P}(\mathcal{X})}$ satisfies $(1/n)l^{(A)}(x^n) < (1/n)\sum_{i=1}^n l^\star(x_i) + 2/n$, giving a normalized description length arbitrarily close to the one-dimensional optimum for n sufficiently large. (In practice, assuming that a large number of symbols are coded, we can always use $El^\star_{\mathcal{P}(\mathcal{X})}(X)$ as the rate for arithmetic coding, neglecting as trivial the $2/n$ term in the above bound.) We deal with floating-point precision issues using the same techniques applied to traditional arithmetic codes.

Example: Again consider the p.m.f. of Table IV(a) and the partition of Fig. 2(a). The interval for the root is $[0, 1)$ with subintervals $[0, 0.21)$ and $[0.21, 1)$ for children "1" $= (a_3, a_6)$ and "2" $= (a_7)$, respectively. Interval $[0.21, 1)$ comprises subintervals $[0.21, 0.3297)$, $[0.3297, 0.5571)$, and $[0.5571, 1)$ for $''21'' = (a_0)$, $''22'' = (a_1)$, and $''23'' = (a_2)$ since

$$\begin{aligned} p^{(A)}((a_0)) &= p^{(A)}((a_7))\frac{Q((a_0))}{Q((a_7)) - q((a_7))} \\ &= 0.79\frac{0.1}{0.79 - 0.13} = 0.1197 \\ p^{(A)}((a_1)) &= p^{(A)}((a_7))\frac{Q((a_1))}{Q((a_7)) - q((a_7))} \\ &= 0.79\frac{0.19}{0.79 - 0.13} = 0.2274 \\ p^{(A)}((a_2)) &= p^{(A)}((a_7))\frac{Q((a_2))}{Q((a_7)) - q((a_7))} \\ &= 0.79\frac{0.37}{0.79 - 0.13} = 0.4428. \end{aligned}$$

Interval $[0.5571, 1)$ comprises subintervals $[0.5571, 0.8437)$ and $[0.8437, 1)$ for "231" $= (a_4)$ and "232" $= (a_5)$ since

$$\begin{aligned} p^{(A)}((a_4)) &= p^{(A)}((a_2))\frac{Q((a_4))}{Q((a_2)) - q((a_2))} \\ &= 0.4428(0.11/(0.37 - 0.2)) = 0.2866 \\ p^{(A)}((a_5)) &= p^{(A)}((a_2))\frac{Q((a_5))}{Q((a_2)) - q((a_2))} \\ &= 0.4428(0.06/(0.37 - 0.2)) = 0.1563. \end{aligned}$$

Fig. 4(b) shows these intervals.

Fig. 4(c) shows the recursive interval refinement procedure for $X^5 = (a_7a_3a_4a_1a_2)$. Symbol $X_1 = a_7$ gives interval $[0.21, 1)$ of length 0.79 (indicated by the bold line). Symbol $X_2 = a_3$ refines the above interval to the interval $[0.21, 0.3759)$ of length $0.21 \cdot 0.79 = 0.1659$. Symbol $X_3 = a_4$ refines that interval to the interval $[0.3024, 0.3500)$ of length $0.2866 \cdot 0.1659 = 0.0475$. This procedure continues, giving final interval $[0.3241, 0.3289)$.

E. Optimal Partitions: Definitions and Properties

The preceding discussion treats matched code design for a given partition $\mathcal{P}(\mathcal{X})$. The partition yielding the best performance remains to be found.

Given a partition $\mathcal{P}(\mathcal{X})$, let $l^{(H)}_{\mathcal{P}(\mathcal{X})}$ and $l^\star_{\mathcal{P}(\mathcal{X})}$ be the Huffman and optimal description lengths, respectively, for $\mathcal{P}(\mathcal{X})$. We say that $\mathcal{P}(\mathcal{X})$ is *optimal for a matched Huffman SISC* on $p(x, y)$ if

$$El^{(H)}_{\mathcal{P}(\mathcal{X})}(X) \le El^{(H)}_{\mathcal{P}'(\mathcal{X})}(X)$$

for any partition $\mathcal{P}'(\mathcal{X})$ for $p(x, y)$ (and, therefore, by Theorems 1 and 3, $El^{(H)}_{\mathcal{P}(\mathcal{X})}(X) \le El(X)$ where l is the description length for any instantaneous lossless SISC on $p(x, y)$). We say that $\mathcal{P}(\mathcal{X})$ is *optimal for a matched arithmetic SISC* on $p(x, y)$ if

$$El^\star_{\mathcal{P}(\mathcal{X})}(X) \le El^\star_{\mathcal{P}'(\mathcal{X})}(X)$$

for any partition $\mathcal{P}'(\mathcal{X})$ for $p(x, y)$ since the arithmetic code approaches the optimal one-dimensional expected rate of Theorem 2 as n (the number of symbols coded) grows.

Some properties of optimal partitions follow. (We give the first two without proof.)

Lemma 2: There exists an optimal partition $\mathcal{P}^\star(\mathcal{X})$ for $p(x, y)$ for which every node except for the root of $\mathcal{P}^\star(\mathcal{X})$ is nonempty and no node except for the root can have exactly one child.

Lemma 3: Let $\mathcal{T}(\boldsymbol{n})$ be an arbitrary node in optimal partition $\mathcal{P}^\star(\mathcal{X})$ for $p(x, y)$, and let $\mathcal{G} = (\boldsymbol{(n)}\colon \mathcal{C}(\boldsymbol{n}))$ be the group with root $\boldsymbol{n}$ and descendants identical to the descendants of $\boldsymbol{n}$ in $\mathcal{P}^\star(\mathcal{X})$. Then, $\boldsymbol{n} = \{x \in \mathcal{G}\colon \{x\} \not\in (\mathcal{G} \cap \{x\}^c)\}$ and $\mathcal{C}(\boldsymbol{n})$ is an optimal partition of $\{x \in \mathcal{G}\colon x \not\in \boldsymbol{n}\}$.

Lemma 4: The optimal partitions for matched Huffman and arithmetic SISCs can differ.

Proof: We give a proof by example. For the p.m.f. of Table IV(a), the optimal partition for a matched Huffman SISC is

$$\{(a_0, a_1), ((a_2, a_7)\colon \{(a_3), (a_5)\}), (a_4, a_6)\}$$

while the optimal partition for a matched arithmetic SISC is

$$\{(a_3, a_6), ((a_7)\colon \{(a_0, a_4), ((a_2)\colon \{(a_1)(a_5)\})\})\}. \quad \square$$

Lemmas 2 and 3 apply to both cases. Lemma 4 results since Huffman codes use true single-symbol coding $(n = 1)$ while arithmetic codes minimize the expected rate when n is large. The rates are related as

$$\begin{aligned} \lim_{n\to\infty} \frac{1}{n} El^\star_{\mathcal{P}^{(A)}(\mathcal{X})}(X^n) &\le El^{(H)}_{\mathcal{P}^{(H)}(\mathcal{X})}(X) \\ &< \lim_{n\to\infty} \frac{1}{n} El^\star_{\mathcal{P}^{(A)}(\mathcal{X})}(X^n) + 1. \end{aligned}$$

F. Partition Design and Complexity

We build an optimal partition for $\mathcal{X}$ by building optimal groups for larger and larger subsets $\mathcal{X}' \subseteq \mathcal{X}$ and testing all legitimate combinations of those groups. Let

$$\mathcal{R}(\mathcal{X}') = \{x \in \mathcal{X}'\colon \{x\} \not\in (\mathcal{X}' \cap \{x\}^c)\}.$$

We eliminate all $\mathcal{X}'$ with $\mathcal{R}(\mathcal{X}') = \phi$ by Lemma 2. By Lemma 3, the optimal group for $\mathcal{X}'$ is

$$\mathcal{G}^\star(\mathcal{X}') = (\mathcal{R}(\mathcal{X}')\colon \mathcal{C}(\mathcal{X}'))$$

where $\mathcal{C}(\mathcal{X}') = \mathcal{P}^\star(\mathcal{X}' \cap \mathcal{R}(\mathcal{X}')^c)$ is the optimal partition on $\mathcal{X}' \cap \mathcal{R}(\mathcal{X}')^c$. Thus, $\mathcal{G}^\star(\{x\}) = (x)$ for any $x \in \mathcal{X}$. For any $\mathcal{X}' \subseteq \mathcal{X}$ with $|\mathcal{R}(\mathcal{X}')| > 0$ and $|\mathcal{X}' \cap \mathcal{R}(\mathcal{X}')^c| > 0$, we find $\mathcal{C}(\mathcal{X}')$ by calculating the expected rate of the matched code for each set of groups of the form

$$\mathcal{C} = \Bigg\{ \mathcal{G}^\star(\mathcal{S}_1), \ldots, \mathcal{G}^\star(\mathcal{S}_K) \colon |\mathcal{R}(\mathcal{S}_k)| > 0 \quad \forall k, \bigcup_{k=1}^{K} \mathcal{S}_k = (\mathcal{X}' \cap \mathcal{R}(\mathcal{X}')^c), \mathcal{S}_i \cap \mathcal{S}_j = \phi, \ \forall i, j \Bigg\}$$

and choosing the one with the best performance.

The number of partitions for which we must design matched codes can be loosely bounded from above by

$$\sum_{k=1}^{|\mathcal{X}|} \binom{|\mathcal{X}|}{k} B_k < 2^{|\mathcal{X}|} B_{|\mathcal{X}|}$$

where $B_m \sim m^{-1/2}[\lambda(m)]^{m+1/2} e^{\lambda(m)-m-1}$ is the number of ways a set of m elements can be partitioned into nonempty subsets and $\lambda(m)\ln[\lambda(m)] = m$ [20].[5]

While the design is expensive, the encoding and decoding complexities for an optimal SISC are comparable to the encoding and decoding complexities of a traditional (single-sender, single-receiver) Huffman or arithmetic code. All are linear in $|\mathcal{X}|$. For Huffman coding, we use a table lookup encoder and a binary tree decoder. The decoder's binary tree labels node $\boldsymbol{n}$ with all $x \in \mathcal{X}$ such that $\gamma_X(x) = \boldsymbol{n}$. Since the decoder knows y, it stops reading bits when it reaches a node $\boldsymbol{n}$ for which there is some $x \in \boldsymbol{n}$ with $p(x, y) > 0$; the decoder outputs that x. Similarly, an arithmetic SISC has encoding and decoding complexities that are comparable to those of traditional arithmetic codes.

III. General Lossless Instantaneous Multiple Access Source Codes

A. Problem Statement, Partition Pairs, and Optimal Matched Codes

We here drop the SISC assumption that Y (or X) can be decoded independently and consider MASC design when it may be necessary to decode the two symbol descriptions together. We replace the SISC partition $\mathcal{P}(\mathcal{X})$ by a pair of partitions $(\mathcal{P}(\mathcal{X}), \mathcal{P}(\mathcal{Y}))$ that describe the prefix and equivalence relationships for $\{\gamma_X(x)\colon x \in \mathcal{X}\}$ and $\{\gamma_Y(y)\colon y \in \mathcal{Y}\}$, respectively.

For an MASC to be *instantaneous*, the decoder must recognize when it reaches the end of $\gamma_X(X)$ and $\gamma_Y(Y)$. We again use tree structures to help us understand the prefix relationships that make instantaneous decoding possible. Let $\mathcal{T}_\mathcal{X}$ and $\mathcal{T}_\mathcal{Y}$ be a pair of binary trees for which each symbol $x \in \mathcal{X}$ resides at the node reached by traversing path $\gamma_X(x)$ from the root of $\mathcal{T}_\mathcal{X}$ and each symbol $y \in \mathcal{Y}$ resides at the node reached by traversing path $\gamma_Y(y)$ from the root of $\mathcal{T}_\mathcal{Y}$. To decode binary strings $\gamma_X(X)\cdots$ and $\gamma_Y(Y)\ldots$, the decoder starts at the roots of $\mathcal{T}_\mathcal{X}$ and $\mathcal{T}_\mathcal{Y}$ and moves down the first few bits of the path $\gamma_X(X)\cdots$ in $\mathcal{T}_\mathcal{X}$ and $\gamma_Y(Y)\cdots$ in $\mathcal{T}_\mathcal{Y}$, in each case stopping when it reaches an occupied node. Let $\boldsymbol{n}_X$ and $\boldsymbol{n}_Y$ denote those occupied nodes, and use $\mathcal{T}_X$ and $\mathcal{T}_Y$ to describe the subtrees comprising, respectively, $\boldsymbol{n}_X$ plus all of its descendants and $\boldsymbol{n}_Y$ plus all of its descendants. For instantaneous coding, at least one of the following conditions must hold.

(A) $X \in \mathcal{T}_X$ or $\boldsymbol{n}_Y$ is a leaf implies that $Y \in \boldsymbol{n}_Y$, and $Y \in \mathcal{T}_Y$ or $\boldsymbol{n}_X$ is a leaf implies that $X \in \boldsymbol{n}_X$;
(B) $X \in \mathcal{T}_X$ implies that $Y \notin \boldsymbol{n}_Y$;
(C) $Y \in \mathcal{T}_Y$ implies that $X \notin \boldsymbol{n}_X$.

Under condition (A), the decoder has reached the end of $\gamma_X(X)$ and $\gamma_Y(Y)$. Under condition (B), the decoder reads the next few bits of $\gamma_Y(Y)\ldots$, traversing the described path in $\mathcal{T}_Y$ to node $\boldsymbol{n}'_Y$ with subtree $\mathcal{T}'_Y$. Condition (C) similarly leads to a new node $\boldsymbol{n}'_X$ and subtree $\mathcal{T}'_X$. If none of these conditions holds, then the decoder is not instantaneous since it cannot determine whether to continue reading one or both of the descriptions. The decoder continues the above procedure until it determines the nodes of $\mathcal{T}_\mathcal{X}$ and $\mathcal{T}_\mathcal{Y}$ where X and Y reside. At each step before the decoding halts, at least one of the conditions (A), (B), or (C) must be satisfied.

For an MASC to be *lossless*, the above procedure's final nodes $\boldsymbol{n}_X$ and $\boldsymbol{n}_Y$ must satisfy $(X, Y) \in \boldsymbol{n}_X \times \boldsymbol{n}_Y$, and for any other $(x', y') \in \boldsymbol{n}_X \times \boldsymbol{n}_Y$, we must have

$$p(X, y') = p(x', Y) = p(x', y') = 0.$$

We define a partition pair $(\mathcal{P}(\mathcal{X}), \mathcal{P}(\mathcal{Y}))$ to be any pair of prefix relationships on $\{\gamma_X(x)\colon x \in \mathcal{X}\}$ and $\{\gamma_Y(y)\colon y \in \mathcal{Y}\}$. The following theorem gives a simple test for determining whether $(\mathcal{P}(\mathcal{X}), \mathcal{P}(\mathcal{Y}))$ gives a lossless instantaneous MASC. Theorem 4 reduces to Lemma 1 when either $\mathcal{P}(\mathcal{X}) = \{(x)\colon x \in \mathcal{X}\}$ or $\mathcal{P}(\mathcal{Y}) = \{(y)\colon y \in \mathcal{Y}\}$. In either of these cases, the general MASC problem reduces to the SISC problem of Section II.

Theorem 4 (MASC Prefix Property): Partition pair $(\mathcal{P}(\mathcal{X}), \mathcal{P}(\mathcal{Y}))$ yields a lossless instantaneous MASC for $p(x, y)$ if and only if for any $(x, y) \neq (x', y')$ with $p(x, y) > 0$ and $p(x', y') > 0$, at least one of $\{\gamma_X(x), \gamma_X(x')\}$ and $\{\gamma_Y(y), \gamma_Y(y')\}$ is prefix free.

Proof: If an MASC is not instantaneous, then there must be a time in the decoding procedure when the decoder reaches nodes $(\boldsymbol{n}_X, \boldsymbol{n}_Y)$ with subtrees $(\mathcal{T}_X, \mathcal{T}_Y)$ but none of conditions (A), (B), or (C) is satisfied. Violating (A), (B), and (C) implies that there must exist a pair $(x, y), (x', y') \in \mathcal{T}_X \times \mathcal{T}_Y$ with $p(x, y) > 0$ and $p(x', y') > 0$ such that either $(x, x') \in \boldsymbol{n}_X \times (\mathcal{T}_X \cap \boldsymbol{n}_X^c)$ and $(y, y') \in \boldsymbol{n}_Y \times \boldsymbol{n}_Y$ or $(x, x') \in \boldsymbol{n}_X \times (\mathcal{T}_X \cap \boldsymbol{n}_X^c)$ and $(y, y') \in \boldsymbol{n}_Y \times (\mathcal{T}_Y \cap \boldsymbol{n}_Y^c)$ or $(x, x') \in \boldsymbol{n}_X \times \boldsymbol{n}_X$ and $(y, y') \in \boldsymbol{n}_Y \times (\mathcal{T}_Y \cap \boldsymbol{n}_Y^c)$. Thus, both $\{\gamma_X(x), \gamma_X(x')\}$ and $\{\gamma_Y(y), \gamma_Y(y')\}$ violate the prefix property. If an MASC is instantaneous but not lossless, then there must be a pair of nodes $(\boldsymbol{n}_X, \boldsymbol{n}_Y)$ and an $(x, y) \neq (x', y')$ for which $(x, y), (x', y') \in \boldsymbol{n}_X \times \boldsymbol{n}_Y$ and both $p(x, y) > 0$ and $p(x', y') > 0$, so that the decoder reaches a node pair instantaneously but cannot decode without loss. In this case, at least one of the following must be true: 1) $x \neq x'$ and $\gamma_X(x) = \gamma_X(x')$ or 2) $y \neq y'$ and

[5] A later optimal design used in [14] uses a bottom-up approach with worst case complexity satisfying the same bound.

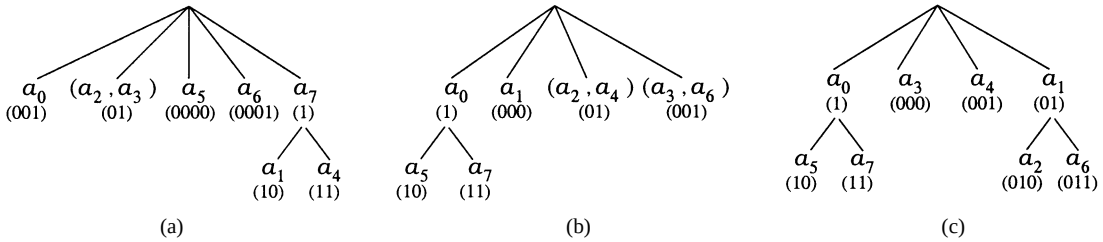


Fig. 5. The partition pair $(\mathcal{P}(\mathcal{X}), \mathcal{P}(\mathcal{Y}))$ with $\mathcal{P}(\mathcal{X})$ shown in (a) and $\mathcal{P}(\mathcal{Y})$ shown in (b) gives a lossless, instantaneous MASC for the p.m.f. in Table IV(a). Replacing $\mathcal{P}(\mathcal{Y})$ with the partition shown in (c) fails to give a lossless, instantaneous MASC for the same p.m.f.

$\gamma_X(y) = \gamma_X(y')$. In either case, the MASC prefix condition is violated.

We now show that if the MASC prefix condition is violated, then we cannot achieve a lossless instantaneous MASC. We begin by building two binary trees, $\mathcal{T}_\mathcal{X}$ and $\mathcal{T}_\mathcal{Y}$. For each $x \in \mathcal{X}$ and $y \in \mathcal{Y}$, we place symbol x at the node reached by path $\gamma_X(x)$ in $\mathcal{T}_\mathcal{X}$ and symbol y at the node reached by path $\gamma_Y(y)$ in $\mathcal{T}_\mathcal{Y}$. If there exists a violation of the MASC prefix condition, then there exists an $(x, y) \neq (x', y')$ for which $p(x, y) > 0$ and $p(x', y') > 0$ and neither $\{\gamma_X(x), \gamma_X(x')\}$ nor $\{\gamma_Y(y), \gamma_Y(y')\}$ is prefix free. If $\boldsymbol{n}_x$, $\boldsymbol{n}_{x'}$, $\boldsymbol{n}_y$, and $\boldsymbol{n}_{y'}$ are the nodes in our tree construction satisfying $x \in \boldsymbol{n}_x$, $x' \in \boldsymbol{n}_{x'}$, $y \in \boldsymbol{n}_y$, and $y' \in \boldsymbol{n}_{y'}$, then one of two cases can occur. If $(\boldsymbol{n}_x, \boldsymbol{n}_y) = (\boldsymbol{n}_{x'}, \boldsymbol{n}_{y'})$, then the example satisfies (A); in this case, the code is not lossless since $\gamma_X(x) = \gamma_X(x')$ and $\gamma_Y(y) = \gamma_Y(y')$ (by construction) and $p(x, y) > 0$ and $p(x', y') > 0$ (by assumption). If $(\boldsymbol{n}_x, \boldsymbol{n}_y) \neq (\boldsymbol{n}_{x'}, \boldsymbol{n}_{y'})$, then either one of $\{\boldsymbol{n}_x, \boldsymbol{n}_{x'}\}$ is the ancestor of the other and $\{\boldsymbol{n}_y, \boldsymbol{n}_{y'}\}$ is not prefix free or one of $\{\boldsymbol{n}_y, \boldsymbol{n}_{y'}\}$ is the ancestor of the other and $\{\boldsymbol{n}_x, \boldsymbol{n}_{x'}\}$ is not prefix free (or both). In this case, none of (A), (B), and (C) is satisfied since the decoder cannot determine whether or not to read beyond the common prefix of $\{\gamma_X(x), \gamma_X(x')\}$ in the description of X or the common prefix of $\{\gamma_Y(y), \gamma_Y(y')\}$ in the description of Y. □

Example: Again consider the p.m.f. in Table IV(a). If we set $\mathcal{P}(\mathcal{X})$ and $\mathcal{P}(\mathcal{Y})$ to be the partitions in Fig. 5(a) and (b), respectively, then there is no pair $(x, y) \neq (x', y')$ with $p(x, y) > 0$ and $p(x', y') > 0$ for which neither $\{\gamma_X(x), \gamma_X(x')\}$ nor $\{\gamma_Y(y), \gamma_Y(y')\}$ is prefix free. Thus, $(\mathcal{P}(\mathcal{X}), \mathcal{P}(\mathcal{Y}))$ gives a lossless, instantaneous code for the p.m.f. in Table IV(a). In contrast, if $\mathcal{P}(\mathcal{X})$ and $\mathcal{P}(\mathcal{Y})$ are the partitions in Fig. 5(a) and (c), respectively, then $p(a_2, a_2) > 0$ and $p(a_3, a_1) > 0$ but $\gamma_X(a_2) = \gamma_X(a_3)$ and $\gamma_Y(a_1) \prec \gamma_Y(a_2)$. Thus, neither $\{\gamma_X(a_2), \gamma_X(a_3)\}$ nor $\{\gamma_Y(a_1), \gamma_Y(a_2)\}$ is prefix free, and the decoder cannot know whether or not to continue reading beyond $\gamma_Y(a_1)$ in decoding the description of Y when it receives $\gamma_X(a_2) = \gamma_X(a_3)$ as its description from X.

Theorem 1 generalizes to show that every lossless, instantaneous MASC is a pair of matched codes for some $(\mathcal{P}(\mathcal{X}), \mathcal{P}(\mathcal{Y}))$ satisfying Theorem 4. Thus, optimal MASC design is equivalent to optimal partition design followed by optimal matched code design. Matched code design for each partition of an MASC is identical to matched code design for the partition of an SISC. Thus, the generalization to optimal matched Huffman and arithmetic codes for any partition pair $(\mathcal{P}(\mathcal{X}), \mathcal{P}(\mathcal{Y}))$ for $p(x, y)$ is immediate. The codewords of an optimal matched Huffman code for the partitions in Fig. 5 appear in parentheses under the nodes of the partition trees.

B. Optimal Partition Properties

Given a partition pair $(\mathcal{P}(\mathcal{X}), \mathcal{P}(\mathcal{Y}))$ that satisfies the MASC prefix condition, $(\mathcal{P}(\mathcal{X}), \mathcal{P}(\mathcal{Y}))$ is *optimal for use in a matched Huffman MASC* on $p(x, y)$ if $(El^{(H)}_{\mathcal{P}(\mathcal{X})}(X), El^{(H)}_{\mathcal{P}(\mathcal{Y})}(Y))$ sits on the lower boundary of the rates achievable by a lossless MASC on alphabet $\mathcal{X} \times \mathcal{Y}$. Similarly, $(\mathcal{P}(\mathcal{X}), \mathcal{P}(\mathcal{Y}))$ is *optimal for use in a matched arithmetic MASC* on $p(x, y)$ if $(El^{\star}_{\mathcal{P}(\mathcal{X})}(X), El^{\star}_{\mathcal{P}(\mathcal{Y})}(Y))$ sits on the lower boundary of $\{(El^{\star}_{\mathcal{P}'(\mathcal{X})}(X), El^{\star}_{\mathcal{P}'(\mathcal{Y})}(Y))$: $(\mathcal{P}'(\mathcal{X}), \mathcal{P}'(\mathcal{Y}))$ are partitions on $\mathcal{X} \times \mathcal{Y}$ for $p(x, y)\}$. Again $l^{(H)}_{\mathcal{P}}$ and $l^{\star}_{\mathcal{P}}$ denote the Huffman and optimal description lengths, respectively, for partition $\mathcal{P}$, and Huffman coding is optimal over all codes on a fixed alphabet. (Mixed codes (e.g., Huffman coding on X and arithmetic coding on Y) are also possible within this framework.)

Lemma 5: For each partition pair $(\mathcal{P}(\mathcal{X}), \mathcal{P}(\mathcal{Y}))$ that achieves performance on the lower boundary of the achievable rate region, there exists a partition pair $(\mathcal{P}^\star(\mathcal{X}), \mathcal{P}^\star(\mathcal{Y}))$ achieving the same rate performance as $(\mathcal{P}(\mathcal{X}), \mathcal{P}(\mathcal{Y}))$ for which every node except for the roots of $\mathcal{P}^\star(\mathcal{X})$ and $\mathcal{P}^\star(\mathcal{Y})$ is nonempty and no node except for the roots can have exactly one child.

Proof: We build $\mathcal{P}^\star(\mathcal{X})$ and $\mathcal{P}^\star(\mathcal{Y})$ by modifying $\mathcal{P}(\mathcal{X})$ and $\mathcal{P}(\mathcal{Y})$ to remove all empty nodes, attaching the node's children directly to its parent. We also remove any nonroot node $\boldsymbol{n}$ that has exactly one child $\boldsymbol{n}1$, combining $\boldsymbol{n}$ and $\boldsymbol{n}1$ to form one-level group $(\boldsymbol{n}, \boldsymbol{n}1)$ with $\{\boldsymbol{n}1k\}_{k=1}^{K(\boldsymbol{n}1)}$ descending directly from $(\boldsymbol{n}, \boldsymbol{n}1)$. Since neither change can increase the code's rate or change the sets of symbols whose descriptions violated the prefix property, we have the desired $(\mathcal{P}^\star(\mathcal{X}), \mathcal{P}^\star(\mathcal{Y}))$ by Theorem 4. □

C. Partition Design and Complexity

For a fixed partition $\mathcal{P}(\mathcal{Y})$ on $\mathcal{Y}$ with matched code γ_Y, we design the optimal partition and matched code γ_X on $\mathcal{X}$ such that (γ_X, γ_Y) satisfy the MASC prefix condition. Traversing through all partitions on $\mathcal{Y}$, we can trace out the lower boundary of achievable rates for MASC.

A very loose bound on the worst case complexity for designing optimal MASC is $C_{\text{MASC}} < 2^{|\mathcal{X}|+|\mathcal{Y}|} B_{|\mathcal{X}|}(|\mathcal{Y}| + 1)!$. The encoding and decoding complexities of the proposed optimal MASCs are linear in the alphabet size and comparable to the corresponding traditional codes.

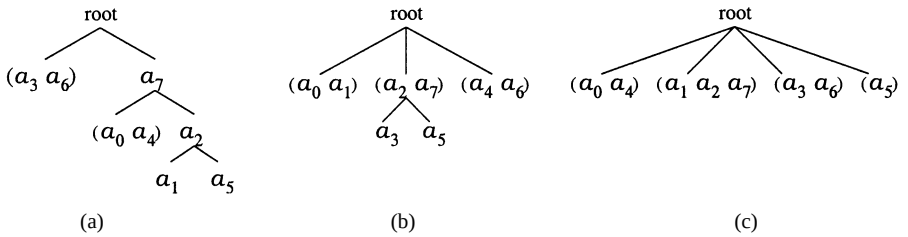


Fig. 6. Partition trees for the p.m.f. from Table IV(a) using (a) optimal arithmetic coding, (b) optimal Huffman coding, (c) arithmetic or Huffman coding with the approach in [3].

IV. Near-Lossless Instantaneous Multiple Access Source Codes

Finally, we generalize from lossless to near-lossless codes. For any fixed small $\epsilon > 0$, we call MASC $((\gamma_X, \gamma_Y), \gamma^{-1})$ a *near-lossless instantaneous MASC for* $P_e \leq \epsilon$ if $((\gamma_X, \gamma_Y), \gamma^{-1})$ yields instantaneous decoding with

$$P_e = Pr(\gamma^{-1}(\gamma_X(X), \gamma_Y(Y)) \neq (X, Y)) \leq \epsilon.$$

Since the code is instantaneous, the decoder builds its reconstruction $(\hat{x}_1, \hat{y}_1)$ of (x_1, y_1) using exactly $|\gamma_X(x_1)|$ bits from $\gamma_X(x_1)\gamma_X(x_2)\gamma_X(x_3)\ldots$ and $|\gamma_Y(y_1)|$ bits from $\gamma_Y(y_1)\gamma_Y(y_2)\gamma_Y(y_3)\ldots$ (without prior knowledge of these lengths). Thus, even when $(\hat{x}_1, \hat{y}_1) \neq (x_1, y_1)$, the decoder correctly determines $|\gamma_X(x_1)|$ and $|\gamma_Y(y_1)|$. This requirement disallows loss of synchronization and error propagation.

Theorem 5 (Near-Lossless MASC Prefix Property): Partition pair $(\mathcal{P}(\mathcal{X}), \mathcal{P}(\mathcal{Y}))$ yields a near-lossless instantaneous MASC for $p(x, y)$ if and only if for any $(x, y) \neq (x', y')$ with $p(x, y) > 0$ and $p(x', y') > 0$, either

(A) at least one of $\{\gamma_X(x), \gamma_X(x')\}$ and $\{\gamma_Y(y), \gamma_Y(y')\}$ is prefix free; or

(B) $\gamma_X(x) = \gamma_X(x')$ and $\gamma_Y(y) = \gamma_Y(y')$.

Proof: Recall that $\gamma_X(x)$ is a *proper* prefix of $\gamma_X(x')$ (written $\gamma_X(x) \prec \gamma_X(x')$) if $\gamma_X(x) \preceq \gamma_X(x')$ and $\gamma_X(x) \neq \gamma_X(x')$. Fix some $(x, y) \neq (x', y')$ with $p(x, y) > 0$ and $p(x', y') > 0$. Under (A), the decoder can instantaneously and losslessly distinguish between (x, y) and (x', y') by Theorem 4. Under (B), we cannot decode losslessly, but there is no ambiguity in how many bits to decode.

If neither (A) nor (B) is satisfied, then there exists an $(x, y) \neq (x', y')$ for which $p(x, y) > 0$ and $p(x', y') > 0$ and either the decoder cannot determine whether to decode $|\gamma_X(x)|$ bits or $|\gamma_X(x')| > |\gamma_X(x)|$ bits because $\gamma_X(x) \prec \gamma_X(x')$ and $\{\gamma_Y(y), \gamma_Y(y')\}$ is not prefix free or the decoder cannot determine whether to decode $|\gamma_Y(y)|$ bits or $|\gamma_Y(y')| > |\gamma_Y(y)|$ bits because $\gamma_Y(y) \prec \gamma_Y(y')$ and $\{\gamma_X(x), \gamma_X(x')\}$ is not prefix free. □

In a near-lossless SISC for X given Y, the prefix condition simplifies to the following: for any $x, x' \in \mathcal{X}$ for which there exists a $y \in \mathcal{Y}$ with $p(x, y) > 0$ and $p(x', y) > 0$, $\gamma_X(x) \prec \gamma_X(x')$ is disallowed (as in lossless coding) but $\gamma_X(x) = \gamma_X(x')$ is allowed. Here $\gamma_X(x) \prec \gamma_X(x')$ would leave the decoder no means of determining whether to decode $|\gamma_X(x)|$ bits or $|\gamma_X(x')|$ bits. However, $\gamma_X(x) = \gamma_X(x')$ allows instantaneous (but not error-free) decoding.

Given a partition pair $(\mathcal{P}(\mathcal{X}), \mathcal{P}(\mathcal{Y}))$ satisfying the near-lossless MASC prefix property, then for any $x \in \boldsymbol{n}_x \in \mathcal{T}(\mathcal{P}(\mathcal{X}))$ and $y \in \boldsymbol{n}_y \in \mathcal{T}(\mathcal{P}(\mathcal{Y}))$ with $p(x, y) > 0$, the optimal decoder gives

$$\gamma^{-1}(\gamma_X(x), \gamma_Y(y)) = \arg \max_{(x', y') \in \boldsymbol{n}_x \times \boldsymbol{n}_y} p(x', y')$$

$$P_e(\mathcal{P}(\mathcal{X}), \mathcal{P}(\mathcal{Y})) = \sum_{(\boldsymbol{n}_\mathcal{X}, \boldsymbol{n}_\mathcal{Y}) \in \mathcal{T}(\mathcal{P}(\mathcal{X})) \times \mathcal{T}(\mathcal{P}(\mathcal{Y}))} \cdot \left[\sum_{(x, y) \in \boldsymbol{n}_\mathcal{X} \times \boldsymbol{n}_\mathcal{Y}} p(x, y) - \max_{(x, y) \in \boldsymbol{n}_\mathcal{X} \times \boldsymbol{n}_\mathcal{Y}} p(x, y) \right].$$

By Theorem 5, we can design a near-lossless MASC by designing a lossless MASC on a reduced alphabet that represents each one-level group by a single symbol. The optimal near-lossless MASC can be found by searching the reduced alphabets that satisfy a given error constraint ϵ. The optimal performance of this one-dimensional code is bounded below by the convex hull of the Slepian–Wolf rate regions on these reduced alphabets.

It is interesting to compare the near-lossless coding approach given above to MASCs based on error correction codes (see, for example, [6], [7], [11]–[13]). The strengths of those algorithms are that they are computationally efficient at high coding dimension n and achieve good coding performance (low error probabilities and rates close to the Slepian–Wolf bound) when the relationship between sources X and Y resembles that between the input and output of the noisy channels for which the error correction code was designed (e.g., the structured sources of [6], [7], [11]–[13]). The weaknesses of these codes are that they do not give instantaneous coding, and they can suffer catastrophic decoding failure due to loss of synchronization when the "errors" between X and Y exceed the code's correction capabilities. In contrast, the strengths of our codes are that they are instantaneous and cannot suffer catastrophic failures. The weaknesses are that code design complexity becomes prohibitive for large coding dimensions, and thus the codes' rate and error probabilities generally fail to meet their asymptotic limits. For example, the smallest error probability that gives a result different from lossless coding for a block-length-n code is $\min\{p^n(x^n, y^n): p^n(x^n, y^n) > 0\}$.

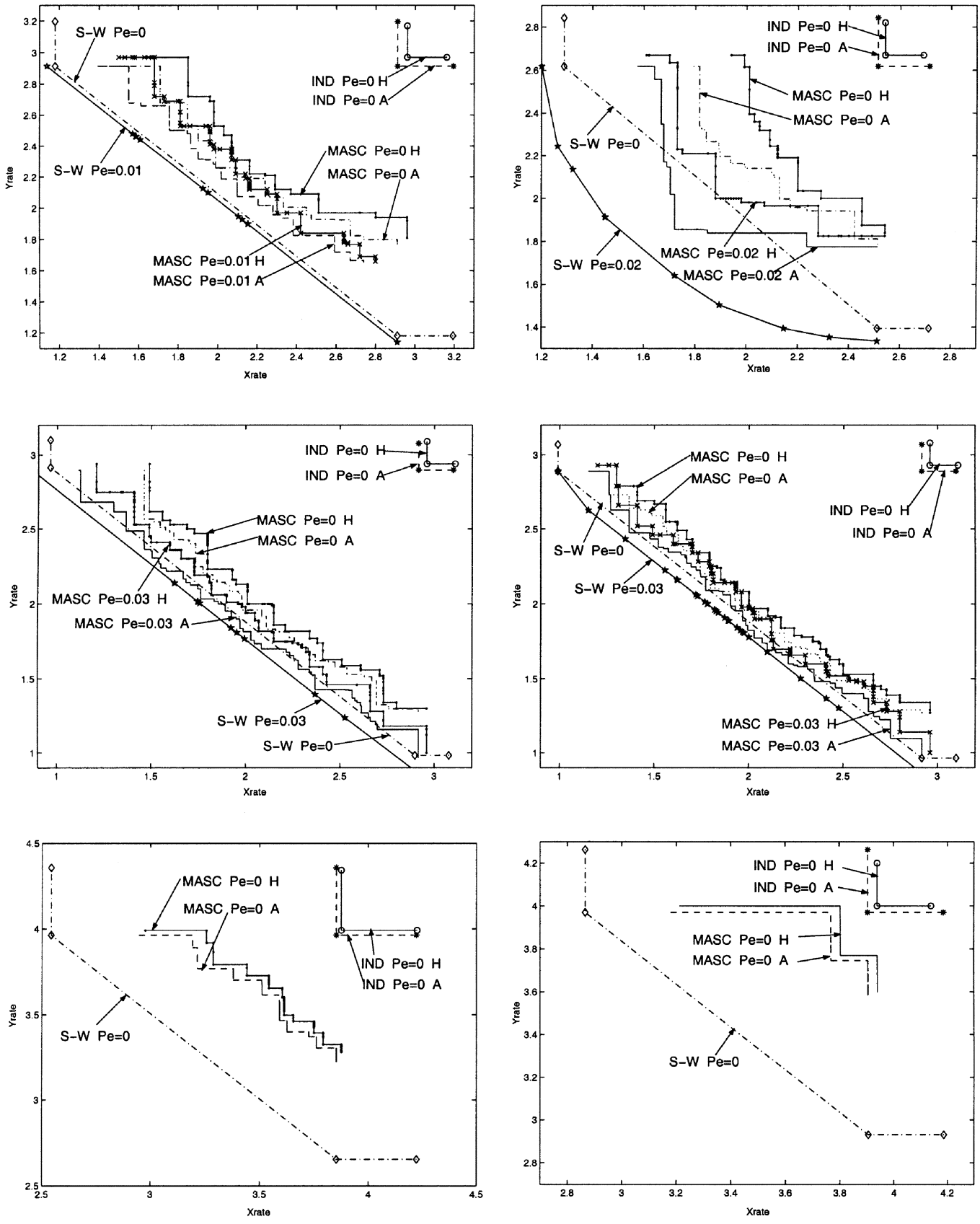
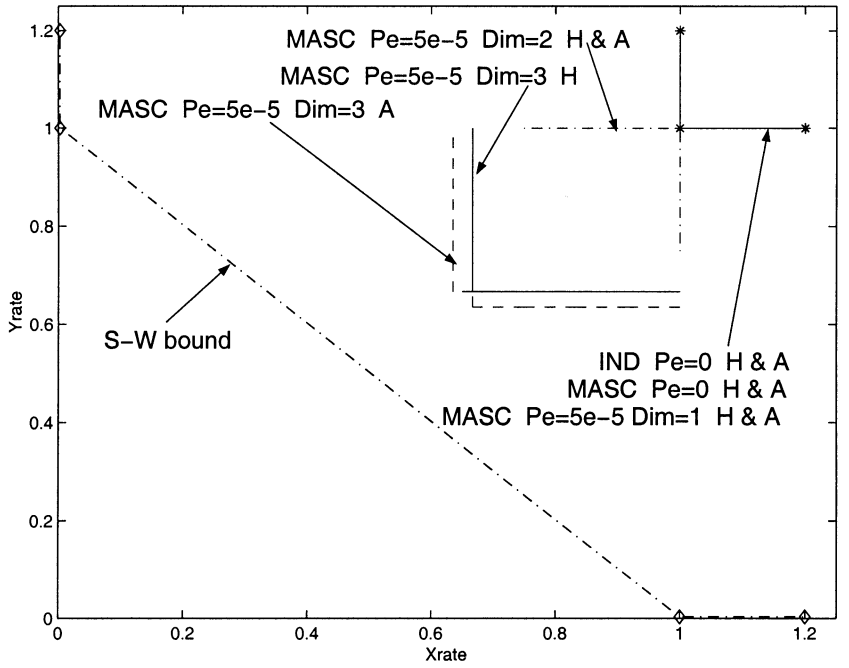
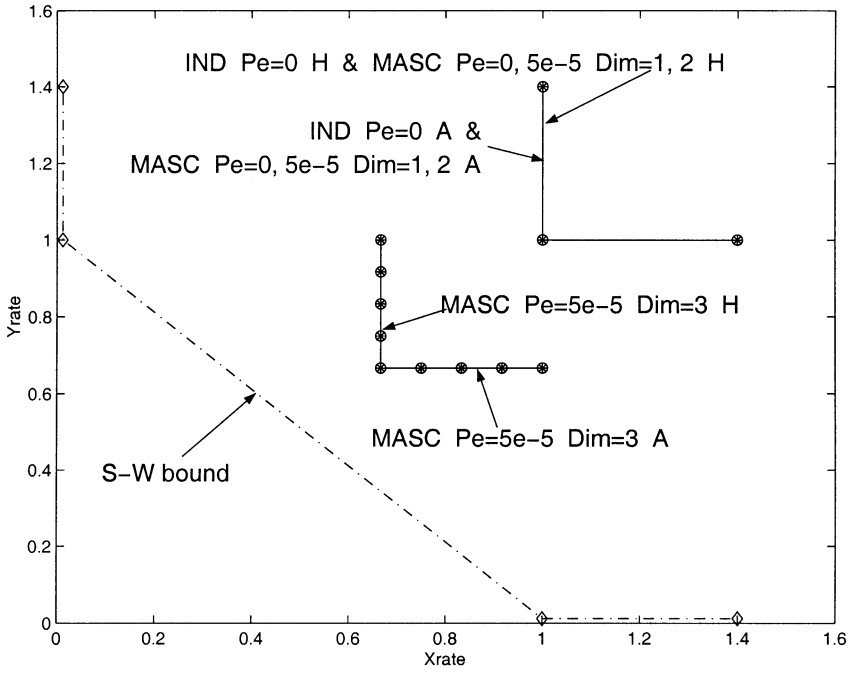


Fig. 7. General dimension-1 lossless and near-lossless MASC results for Table IV(a) (top left), (b) (top right), (c) (middle left), (d) (middle right), Table V(a) (bottom left), and (b) (bottom right). (H: Huffman code; A: arithmetic code; S-W, $P_e = 0$: Slepian–Wolf bound; S-W, $P_e = \epsilon$: bound for near-lossless MASC in Section IV.)



(a)



(b)

Fig. 8. Performance of near-lossless MASC as a function of coding dimension (Dim), for (a) $\alpha = \beta = 0.0002$ and (b) $\alpha = 0.002$, $\beta = 0.0002$. (H: Huffman code; A: arithmetic code).

V. Polynomial Multiple Access Source Coding Algorithms

Since the complexity of optimal partition design is high, we next consider fast partition design algorithm. The approach is conceptually similar to iterative descent algorithms like the generalized Lloyd algorithm. We pick an initial condition at random and then perform a sequence of constrained optimizations. To avoid local minimality problems, we consider several initial starting points (see [18] for alternative methods). The following description focuses on the design of $\mathcal{P}(\mathcal{X})$ for lossless coding. Generalization to lossy coding appears in [17], [18]. The results apply both to SISC design and MASC design under the assumption of a fixed, known $\mathcal{P}(\mathcal{Y})$. The only difference lies in the prefix property and group definitions.

A. Optimizing Order-Constrained Partitions

Order alphabet $\mathcal{X}$ as $\mathcal{O} = \{x_1, x_2, \ldots, x_N\}$, where $N = |\mathcal{X}|$ and $i < j$ implies x_i precedes x_j in ordering $\mathcal{O}$. An *order-*

TABLE I
LOSSLESS SISC RESULTS FOR THE p.m.f.'s OF TABLES IV AND V. HERE $[H(X),\ R'_{SI,\,A}(X),\ R^{\star}_{SI,\,A}(X)]$ AND $[R_H(X),\ R'_{SI,\,H}(X),\ R^{\star}_{SI,\,H}(X)]$ DENOTE THE OPTIMAL AND HUFFMAN RESULTS, RESPECTIVELY, FOR [TRADITIONAL, SISC [3], OPTIMAL SISC] CODING ON X WHEN Y IS GIVEN AS SIDE INFORMATION TO THE DECODER

Table	$H(X)$	$R'_{SI,A}(X)$	$R^{\star}_{SI,A}(X)$	$R_H(X)$	$R'_{SI,H}(X)$	$R^{\star}_{SI,H}(X)$
IV(a)	2.91075	1.67976	1.53582	2.96	1.75	1.67
IV(b)	2.51160	1.8201	1.79381	2.54	1.94	1.94
IV(c)	2.91623	1.5071	1.46162	2.96	1.56	1.49
IV(d)	2.91623	1.2784	1.15161	2.96	1.46	1.2
V(a)	3.85278	3.04098	2.94631	3.8764	3.07865	2.97472
V(b)	3.90508	3.27019	3.17971	3.93979	3.31152	3.21204

TABLE II
COMPARING THE RUNNING TIME OF OPTIMAL DESIGN APPROACHES

Table	IV(a)	IV(b)	IV(c)	IV(d)	V(a)	V(b)
SISC on X (A)	1	1	1	1	1	1
SISC on X (B)	13.056	11.40	20.56	3.681	1.26×10^5	1.36×10^6
SISC on Y (A)	1	1	1	1	1	1
SISC on Y (B)	7.672	15.20	3.664	3.661	8050.32	89145.32

TABLE III
FAST SISC DESIGN RESULTS FOR $p((x_1,\ x_2),\ (y_1,\ y_2)) = p(x_1,\ y_1)p(x_2,\ y_2)$ USING THE p.m.f.'s OF TABLES IV AND V. HERE, $[H(X),\ R^f_A(X)]$ AND $[R_H(X),\ R^f_H(X)]$ DENOTE THE ARITHMETIC AND HUFFMAN RESULTS, RESPECTIVELY, FOR [TRADITIONAL, FAST SISC] CODING OF X

Table	$H(X)$	$R^f_A(X)$	$R_H(X)$	$R^f_H(X)$
IV(a)	5.8215	3.10657	5.8657	3.2286
IV(b)	5.0232	3.58694	5.0536	3.6861
IV(c)	5.83246	2.9264	5.8697	2.9573
IV(d)	5.83246	2.36113	5.8697	2.4809
V(a)	7.70557	5.99109	7.73585	6.13694
V(b)	7.81016	6.46308	7.83855	6.5928

constrained partition for ordering $\mathcal{O}$ is any $\mathcal{P}(\mathcal{O})$ satisfying the following.

1) Any one-level group $\boldsymbol{n} \in \mathcal{T}(\mathcal{P}(\mathcal{O}))$ is a sequence of adjacent elements in $\mathcal{O}$.
2) Define the position of a group to span from its first member to its last, then the root and descendants of every multilevel group $\mathcal{T}(\mathcal{G}) \in \mathcal{T}(\mathcal{P}(\mathcal{O}))$ must hold adjacent positions in $\mathcal{O}$.

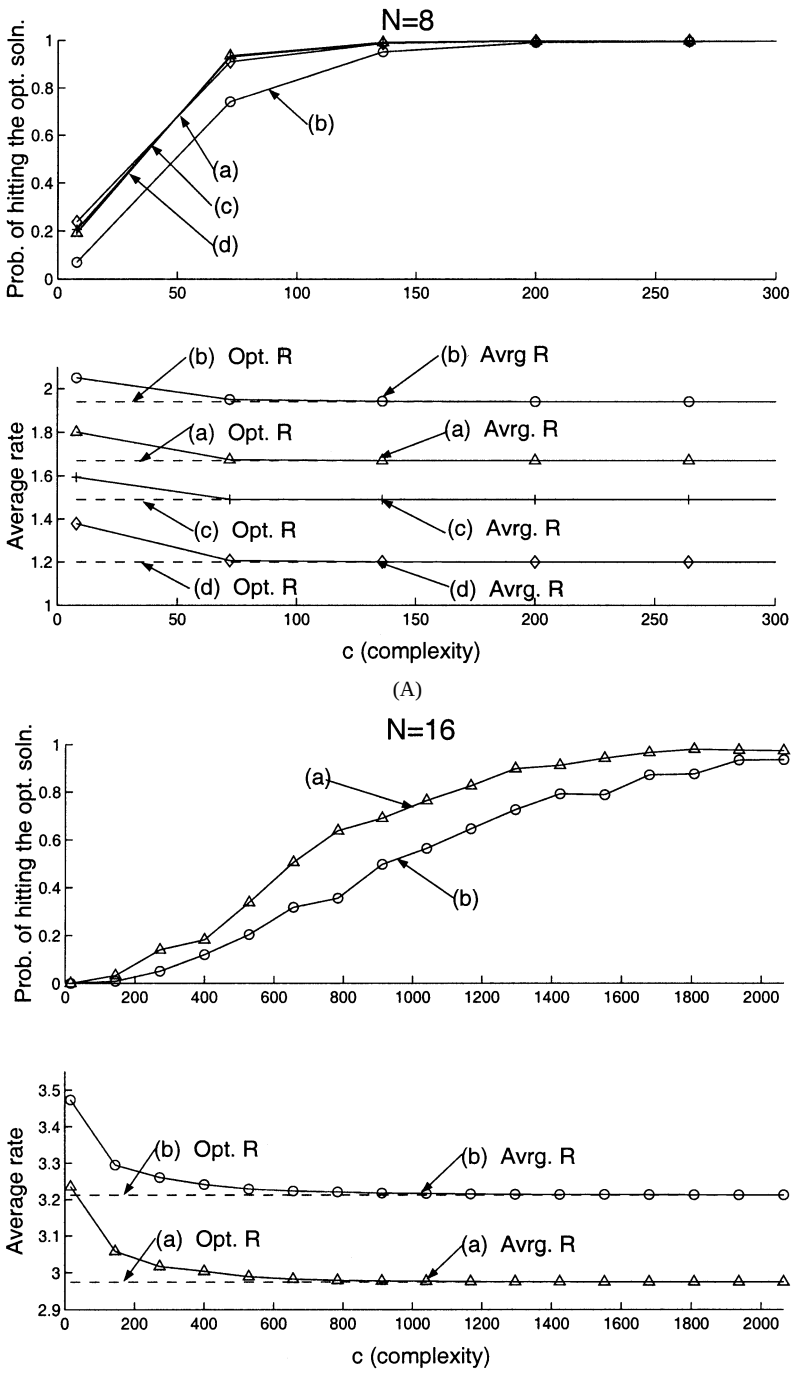
Any $\mathcal{P}(\mathcal{X})$ is an order-constrained partition on a variety of orderings (e.g., the ordering given by a depth-first search [21] of $\mathcal{T}(\mathcal{P}(\mathcal{X}))$). The *optimal* order-constrained partition for $\mathcal{O}$ is the one whose matched code achieves the minimal rate. The *optimal ordering* on $\mathcal{X}$ is the one whose optimal order-constrained partition achieves the minimal rate. Given a particular ordering, Theorem 6 demonstrates that (globally) optimal order-constrained partition design can be achieved in polynomial time. The proof appears in the Appendix.

Theorem 6: The worst case complexity in constructing the optimal order-constrained partition and matched code for a given ordering $\{x_1, \ldots, x_N\}$ is $O(N^4)$.

B. The Iterative Descent Algorithm

Given the above algorithm for designing an optimal order-constrained partition for any ordering, the code design algorithm proceeds as follows. We initialize the algorithm by choosing an ordering $\mathcal{O}_1$ at random. At each time $i \geq 1$, we design the optimal order-constrained partition $\mathcal{P}(\mathcal{O}_i)$, and then choose the new ordering $\mathcal{O}_{i+1}$ by performing a single, randomly chosen permutation on $\mathcal{T}(\mathcal{P}(\mathcal{O}_i))$. Allowed permutations include switching the order of the descendants of any internal node in $\mathcal{T}(\mathcal{P}(\mathcal{O}_i))$ or switching the order of a root with its children. Let $R(\mathcal{O})$ denote the optimal rate of $\mathcal{O}$. Since $\mathcal{P}(\mathcal{O}_i)$ is also an order-constrained partition for $\mathcal{O}_{i+1}$, $R(\mathcal{O}_{i+1}) \leq R(\mathcal{O}_i)$.

In our experiments, we choose a new initial condition each time $R(\mathcal{O}_i)$ remains unchanged for several iterations and the best ordering the algorithm outputs after c iterations is $\min_i\{R(\mathcal{O}_i)\}$ (which is not necessarily equal to $R(\mathcal{O}_c)$). In our examples, $c = N^3$ (corresponding to a complexity of $O(N^7)$) gives good experimental results. Like the Lloyd algorithm, however, this design strategy cannot guarantee a good solution, but seems to work well in practice.



(A)

(B)

Fig. 9. Performance as a function of complexity for the polynomial-time design algorithm on the p.m.f.'s of (A) Table IV and (B) Table V.

VI. Experimental Results

A. Optimal Algorithms

This section shows optimal coding rates for lossless SISCs, lossless MASCs, and near-lossless MASCs for the p.m.f.'s of Tables IV and V in the Appendix. We achieve these results by building the optimal partitions and matched codes for each scenario using the algorithms discussed in Sections II–IV. Both Huffman and arithmetic coding rates are included.

Table I gives SISC results. As an example, for the p.m.f. in Fig. 4(a), the rate achievable in coding X using side information Y is approximately half that of an ordinary Huffman code and 90% that of [3]; the corresponding partition trees appear in Fig. 6. The number of partitions needed to be tested to get the optimal rates ranges from $2|\mathcal{X}|^2$ to $|\mathcal{X}|^3$ for these p.m.f. examples.

Fig. 7 shows general lossless and near-lossless MASC results compared with the corresponding bounds and the independent coding results. The optimal lossless MASC gives significant performance improvement over independent coding of

TABLE IV
SAMPLE p.m.f.'s ON ALPHABET $\mathcal{X} \times \mathcal{Y}$ WITH $\mathcal{X} = \mathcal{Y} = \{a_0, a_1, \ldots, a_6, a_7\}$

(a)

x\ y	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
a_0	.04	0	.04	.02	0	0	0	0
a_1	0	.04	0	0	.05	.1	0	0
a_2	.15	0	.05	0	0	0	0	0
a_3	0	.05	0	.06	0	0	0	0
a_4	0	.06	0	0	.05	0	0	0
a_5	0	0	0	.01	.02	.03	0	0
a_6	0	0	.01	0	0	.06	.02	.01
a_7	0	0	0	0	0	0	.05	.08

(b)

x\ y	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
a_0	.1	0	.1	.1	0	0	0	0
a_1	.06	.04	0	0	.05	0	.1	0
a_2	.15	0	.05	0	0	0	0	0
a_3	0	.05	0	.05	0	0	0	0
a_4	0	.04	0	0	.02	0	0	0
a_5	0	0	0	.02	.01	.01	0	0
a_6	0	0	.01	0	0	0	.015	.005
a_7	0	0	0	0	0	.01	0	.01

(c)

x\ y	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
a_0	.06	0	0	0	.04	0	0	0
a_1	0	0	.05	.04	0	0	0	0
a_2	0	0	.05	0	0	0	0	.15
a_3	.06	0	0	0	.05	0	0	0
a_4	.05	0	0	0	0	.06	0	0
a_5	0	0	0	0	0	0	.07	0
a_6	0	.06	0	.1	0	0	0	.03
a_7	0	0	0	0	.08	0	.05	0

(d)

x\ y	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7
a_0	.06	.04	0	0	0	0	0	0
a_1	0	.04	.05	0	0	0	0	0
a_2	0	0	.05	.15	0	0	0	0
a_3	0	0	0	.06	.05	0	0	0
a_4	0	0	0	0	.05	.06	0	0
a_5	0	0	0	0	0	0	.07	0
a_6	0	0	0	0	0	.1	.06	.03
a_7	0	0	0	0	0	0	.05	.08

X and Y. For the example of Table IV(a), near-lossless coding with error probability 0.01 gives big improvements over lossless coding. The number of partitions that needed to be tested to trace out the given rate regions are bounded above by $2|\mathcal{X}|^4$, $|\mathcal{X}|^4$, $2|\mathcal{X}|^6$, $2|\mathcal{X}|^6$, $2|\mathcal{X}|^4$, and $|\mathcal{X}|^3/2$, respectively, for these six p.m.f.'s.

Fig. 8 shows the effect of the coding dimension on the achievable rate for a fixed error probability. We use the following p.m.f. Let $\mathcal{X} = \mathcal{Y} = \{0, 1\}$ and $p_Y(1) = 0.5$, $p_{X|Y}(0|0) = \alpha$, and $p_{X|Y}(1|1) = \beta$ for some $0 \leq \alpha, \beta < 0.5$. Fig. 8 shows the near-lossless MASC performance for $P_e \leq 5 \times 10^{-5}$ at parameter values $\alpha = \beta = 0.0002$ and $\alpha = 0.002$, $\beta = 0.0002$. As the coding dimension increases, the achievable rate region improves. The number of partitions needed to trace out the rate regions is bounded above by 2^8 for dimension-3, $\epsilon = 5e - 5$ near-lossless MASCs.

Finally, in Table II, we compare the running time of (A) the optimal SISC design of Section II with (B) a later optimal design used in [14]. All experiments are run under identical conditions. Results are normalized to the running time of (A). While no general results are available, the comparison of (A) and (B) demonstrates the existence of examples where the more structured search presented in Section II reduces complexity relative to [14] very significantly.

B. Polynomial-Time Design Algorithms

In this subsection, we present experimental results for the suboptimal design algorithm described in Section V. We again use the p.m.f.'s from Tables IV and V. We show the performance of our algorithm as a function of its complexity. Since the algorithm involves random ordering choices, we run each experiment 500 times. We measure the complexity c of a trial by the number of orderings tested. We measure the performance of a trial both by the fraction of trials in which the algorithm finds the optimal code and by the average (over trials) of the code's rate at the end of a trial.

Fig. 9 shows the resulting Huffman coding performance. In these experiments, the performance improves greatly as c increases from N to N^2, and tends to approach optimality as c approaches N^3. In the average rate measurements we see good performance by $c = 2N^2$ even when the code fails to find the optimal solution.

The previous examples use small alphabet sizes $N \in \{8, 16\}$; we next construct two larger alphabet examples by setting

$$p((x_1, x_2), (y_1, y_2)) = p(x_1, y_1)p(x_2, y_2)$$

for the p.m.f.'s in Tables IV and V. (We assume the independence of (X_1, Y_1) and (X_2, Y_2).) Table III gives the fast algorithm's results using a single trial and $c = 10^4$.

VII. SUMMARY

This paper demonstrates that the optimal lossless and near-lossless MASC design problems can be broken into two subproblems: partition design and matched code design. The partition of an MASC describes the prefix and equivalence relationships for the code's binary descriptions. We give necessary and sufficient conditions on these partitions for instantaneous and lossless or near-lossless decoding and describe a variety of properties of the optimal partition that decrease the complexity associated with optimal partition design. We demonstrate the relationship between optimal matched codes and traditional (single-sender, single-receiver) source codes and use this relationship to give optimal matched code design algorithms. When combined, these results characterize lossless and near-lossless SISCs and MASCs and yield a means of searching for the optimal codes of those types for an arbitrary source p.m.f. $p(x, y)$. Experimental results based on

TABLE V
SAMPLE p.m.f.s $p(x, y)$ ON ALPHABET $\mathcal{X} \times \mathcal{Y}$ WITH $\mathcal{X} = \mathcal{Y} = \{a_0, a_1, \ldots, a_{15}\}$

(a) $356 \cdot p(x, y)$

x\ y	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_0	7	3	0	0	6	4	0	0	3	2	0	0	0	0	0	0
a_1	0	9	0	0	4	3	0	0	0	3	0	0	0	0	0	0
a_2	0	6	0	0	0	6	0	0	0	6	0	0	0	0	0	0
a_3	0	0	3	2	0	0	7	3	0	0	3	3	0	0	0	0
a_4	1	0	2	0	0	0	1	1	0	0	2	0	1	1	0	0
a_5	3	3	0	6	3	3	0	0	3	3	0	6	3	3	0	0
a_6	0	0	6	0	0	0	9	6	0	0	6	0	0	0	6	6
a_7	3	3	0	3	3	3	0	0	3	3	0	3	3	3	0	0
a_8	4	0	2	0	2	0	2	1	2	0	2	0	2	0	2	0
a_9	0	2	0	1	0	2	0	0	0	2	0	1	0	2	0	1
a_{10}	3	0	3	6	0	0	3	6	3	0	8	6	0	0	3	6
a_{11}	0	3	0	6	0	0	0	6	0	3	0	0	0	0	0	6
a_{12}	0	0	0	0	3	2	0	0	0	2	0	0	5	2	0	0
a_{13}	0	0	0	0	0	0	2	2	3	0	2	2	0	0	2	2
a_{14}	0	0	0	0	4	2	0	0	0	2	0	0	4	2	0	0
a_{15}	0	0	0	0	0	0	0	3	3	0	0	3	0	0	0	6

(b) $382 \cdot p(x, y)$

x\ y	a_0	a_1	a_2	a_3	a_4	a_5	a_6	a_7	a_8	a_9	a_{10}	a_{11}	a_{12}	a_{13}	a_{14}	a_{15}
a_0	3	2	0	0	3	2	0	0	3	2	0	0	3	2	0	0
a_1	0	3	0	1	4	3	2	0	0	3	0	1	4	3	2	1
a_2	0	6	0	0	0	6	0	0	0	6	0	0	0	6	0	0
a_3	0	0	3	3	0	0	3	3	0	0	3	3	0	0	3	3
a_4	1	0	2	2	1	1	0	0	0	0	2	2	1	1	0	0
a_5	3	3	0	6	3	3	0	0	3	3	0	6	3	3	0	0
a_6	0	0	6	0	0	0	6	6	0	0	6	0	0	0	6	6
a_7	3	3	0	3	3	3	0	0	3	3	0	3	3	3	0	0
a_8	2	0	2	0	2	0	2	1	2	0	2	0	2	0	2	0
a_9	0	2	0	1	0	2	0	0	0	2	0	1	0	2	0	1
a_{10}	3	0	3	6	0	0	3	6	3	0	3	6	0	0	3	6
a_{11}	0	3	0	6	0	0	0	6	0	3	0	6	0	0	0	6
a_{12}	0	2	0	0	3	2	0	0	0	2	0	1	3	2	0	1
a_{13}	3	0	2	2	0	0	2	2	3	0	2	2	0	0	2	2
a_{14}	0	2	0	0	4	2	0	0	0	2	0	0	4	2	0	0
a_{15}	3	0	0	3	0	0	0	3	3	0	0	3	0	0	0	3

this algorithm are consistent with the theory of MASCs and demonstrate its feasibility in optimal code design on small alphabets. While optimal MASC code design is NP-hard, we provide polynomial-time algorithms which approximate the optimal design for general p.m.f.'s.

APPENDIX

A. Proof of Theorem 6

Theorem 6: The worst case complexity in constructing the optimal order-constrained partition and matched code for a given ordering $\{x_1, \ldots, x_N\}$ is $O(N^4)$.

Proof: By construction: For any $i \le j$, let

$$\mathcal{G}[i, j] = (\mathcal{R}[i, j]: \mathcal{C}[i, j])$$

be the optimal order-constrained group with members $\{x_i, \ldots, x_j\}$. We wish to find the optimal order-constrained partition $\mathcal{G}[1, N]$ given only

$$\mathcal{G}[i, i] = ((x_i): \{\}) = (x_i)$$

for each $i \in \{1, \ldots, N\}$. We use a dynamic programming approach to build $\mathcal{G}[i, j]$ for larger and larger subsets of the source alphabet.

We find group $G[i, k]$ by considering combinations of $G[i, j]$ and $G[j + 1, k]$ for all $j \in \{i, \ldots, k - 1\}$. Let $w[i, j]$ denote the rate associated with the optimal group $\mathcal{G}[i, j]$. Then

$$w[i, i] = 0, \qquad \text{for all } i \in \{1, \ldots, N\}.$$

For any j, let $c[i, j, k]$ describe the relationship of the structures of $G[i, j]$ and $G[j + 1, k]$. If Γ is the set of all legitimate order-constrained multilevel groups. Let Γ be the set of all legitimate order-constrained multilevel groups

$$P[i, j] = \sum_{\ell=i}^{j} p_X(x_\ell)$$

and

$$H(p) = -p \log p - (1 - p) \log(1 - p).$$

Then

$$c[i,\,j,\,k] = \begin{cases} 0, & \text{if } w[i,\,j] = 0 \text{ and } ((\mathcal{G}[i,\,j] \cup \mathcal{R}[j+1,\,k])\text{: } \mathcal{C}[j+1,\,k]) \in \Gamma \\ 1, & \text{if } w[i,\,j] > 0,\ w[j+1,\,k] = 0, \text{ and } ((\mathcal{G}[j+1,\,k] \cup \mathcal{R}[i,\,j])\text{: } \mathcal{C}[i,\,j]) \in \Gamma \\ 2, & \text{otherwise (in this case, we get group } (()\text{: } \{\mathcal{G}[i,\,j],\,\mathcal{G}[j+1,\,k]\}) \in \Gamma). \end{cases}$$

and

$$w[i,\,j,\,k] = \begin{cases} w[j+1,\,k], & \text{if } c[i,\,j,\,k] = 0 \\ w[i,\,j], & \text{if } c[i,\,j,\,k] = 1 \\ w[i,\,j] + w[j+1,\,k] + P[i,\,k], & \text{if } c[i,\,j,\,k] = 2 \text{ in Huffman coding} \\ w[i,\,j] + w[j+1,\,k] + P[i,\,k]H(P[i,\,j]/P[i,\,k]), & \text{if } c[i,\,j,\,k] = 2 \text{ in arithmetic coding.} \end{cases}$$

Finally, if

$$j^\star = j^\star[i,\,k] = \arg \min_{j \in \{i, i+1, \ldots, k-1\}} w[i,\,j,\,k]$$

then $w[i,\,k] = w[i,\,j^\star,\,k]$ and

$$\mathcal{G}[i,\,k] = \begin{cases} ((\mathcal{G}[i,\,j^\star] \cup \mathcal{R}[j^\star+1,\,k])\text{: } \mathcal{C}[j^\star+1,\,k]), & \text{if } c[i,\,j^\star,\,k] = 0 \\ ((\mathcal{G}[j^\star+1,\,k] \cup \mathcal{R}[i,\,j^\star])\text{: } \mathcal{C}[i,\,j^\star]), & \text{if } c[i,\,j^\star,\,k] = 1 \\ (()\text{: } \{\mathcal{G}[i,\,j^\star],\,\mathcal{G}[j^\star+1,\,k]\}), & \text{if } c[i,\,j^\star,\,k] = 2. \end{cases}$$

When the procedure is complete, $\mathcal{G}[1,\,N]$ is the optimal order-constrained partition on ordering $\{x_1, \ldots, x_N\}$ and $w[1,\,N]$ is its expected description length.

The number of operations required to calculate $w[i,\,j,\,k]$ dominates the complexity of the above algorithm. There are

$$\sum_{i=1}^{N-1} \sum_{j=i}^{N-1} \sum_{k=j+1}^{N} 1 = O(N^3)$$

values of $w[i,\,j,\,k]$. In the worst case, calculating $w[i,\,j,\,k]$ requires checking the confusability of one-level groups of size $j-i+1$ and $k-j$ to find $c[i,\,j,\,k]$; since the confusability of every subset of these groups has been compared in a previous step, the new comparison requires at most $\min\{j-i+1,\,k-j\} \le (k-i)/2$ new comparisons. Thus, the complexity is

$$\sum_{i=1}^{N-1} \sum_{j=i}^{N-1} \sum_{k=j+1}^{N} (k-i)/2 = O(N^4). \qquad \square$$

B. Joint Probability Examples

Tables IV and V give four 8×8 and two 16×16 p.m.f. examples.

Acknowledgment

The authors gratefully acknowledge the detailed critiques and many helpful comments from the anonymous reviewers.

References

[1] D. Slepian and J. K. Wolf, "Noiseless coding of correlated information sources," *IEEE Trans. Inform. Theory*, vol. IT-19, pp. 471–480, July 1973.

[2] H. S. Witsenhausen, "The zero-error side information problem and chromatic numbers," *IEEE Trans. Inform. Theory*, vol. IT-22, pp. 592–593, 1976.

[3] A. Kh. Al Jabri and S. Al-Issa, "Zero-error codes for correlated information sources," in *Proc. Cryptography*, Cirencester, UK, Dec. 1997, pp. 17–22.

[4] Y. Yan and T. Berger, "On instantaneous codes for zero-error coding of two correlated sources," in *Proc. IEEE Int. Symp. Information Theory*, Sorrento, Italy, June 2000, p. 344.

[5] N. Alon and A. Orlitsky, "Source coding and graph entropies," *IEEE Trans. Inform. Theory*, vol. 42, pp. 1329–1339, Sept. 1996.

[6] S. S. Pradhan and K. Ramchandran, "Distributed source coding using syndromes (DISCUS) design and construction," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 1999, pp. 158–167.

[7] ——, "Distributed source coding: Symmetric rates and pplications to sensor networks," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 2000, pp. 363–372.

[8] T. J. Flynn and R. M. Gray, "Encoding of correlated observations," *IEEE Trans. Inform. Theory*, vol. IT-33, pp. 773–787, Nov. 1987.

[9] R. Zamir and S. Shamai (Shitz), "Nested linear/lattice codes for Wyner–Ziv encoding," in *Proc. Information Theory Workshop*, Killarney, Ireland, June 1998, pp. 92–93.

[10] M. Fleming, Q. Zhao, and M. Effros, "Network vector quantization," *IEEE Trans. Inform. Theory*, submitted for publication.

[11] J. Bajcsy and P. Mitran, "Coding for the Slepian–Wolf problem with turbo codes," in *Proc. GLOBECOM*, Nov. 2001, pp. 1400–1404.

[12] A. Aaron and B. Girod, "Compression with side information using turbo codes," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 2002, pp. 252–261.

[13] Y. Zhao and J. Garcia-Frias, "Data compression of correlated nonbinary sources using punctured turbo codes," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 2002, pp. 242–251.

[14] P. Koulgi, E. Tuncel, S. Regunathan, and K. Rose, "Minimum redundancy zero-error source coding with side information," in *Proc. IEEE Int. Symp. Information Theory*, Washington, DC, June 2001, p. 282.

[15] Q. Zhao and M. Effros, "Optimal code design for lossless and near-lossless source coding in multiple access networks," in *Proc. Data Compression Conf.*, Snowbird, UT, Mar. 2001, pp. 263–272.

[16] ——, "Lossless source coding for multiple access networks," in *Proc. IEEE Int. Symp. Information Theory*, Washington, DC, June 2001, p. 285.

[17] Q. Zhao, S. Jaggi, and M. Effros, "Side information source coding: Low complexity design and source independence," in *Conf. Rec. 36th Asilomar Conf. Signals, Systems, and Computers*, Invited Paper, Pacific Grove, CA, Nov. 2002.

[18] Q. Zhao and M. Effros, "Low complexity code design for lossless and near—lossless side information source codes," presented at the Data Compression Conference 2003.

[19] P. Koulgi, E. Tuncel, S. Regunathan, and K. Rose, "On zero-error source coding with decoder side information," *IEEE Trans. Inform. Theory*, to be published.

[20] L. Lovász, *Combinatorial Problems and Exercises*, 2nd ed. Amsterdam, The Netherlands: North-Holland, 1993.

[21] S. Even, *Graph Algorithms*. Potomac, MD: Computer Science, 1979.